

A DUAL BASED PROCEDURE FOR A SPECIAL CASE OF TRANSSHIPMENT PROBLEM

**A Thesis submitted
in Partial fulfillment of the Requirements
for the degree of**

MASTER OF TECHNOLOGY

by

**Amit Kumar Saxena
(97110401) M.Tech. [IME]**

to the

**DEPARTMENT OF INDUSTRIAL AND MANAGEMENT ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR
December, 1998.**

17 MAY 1999 TIME
CENTRAL LIBRARY
I I T., KANPUR
No. A 127913



A127913

CERTIFICATE

This is to certify that the present work entitled “**A Dual Based Procedure for Transshipment Problem**” has been carried out by Mr Amit Kumar Saxena under my supervision and that it has not been submitted elsewhere for a degree

December, 1998



Dr. R. R. K. Sharma

Associate Professor

Department of Industrial and Management Engineering

Indian Institute of Technology

Kanpur - 208016 , INDIA

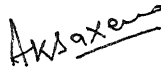
ACKNOWLEDGEMENT

I take this opportunity to express my sincere gratitude towards Dr. R. R. K. Sharma whose able guidance helped me in completing my thesis. At various stages, he guided me to the relevant literature , offered critical comments and helped me in conceptualization

I am thankful to all faculty members of IME department , who exposed us to the excellent academic environment of IIT Kanpur

Finally I would like to thank all my friends of IME family who made my stay in the campus pleasant and memorable one

Date · 22-12-98


(Amit Kumar Saxena)

ABSTRACT

In this thesis we have given a dual based procedure for solving the transshipment problem . The procedure given by us in this thesis runs in $O (c n^3)$ number of steps . We found that this procedure reaches 97 % closer to the optimal solution on an average . Thus a transshipment problem can reach very close to optimal solution and after that it can leave it to conventional $O (n^3 \log (n))$ procedures to reach the optimal solution . Our contribution is to reach very close to optimal solution in attractive computational time

Chapter No.	Description	Page No.
•	<i>Acknowledgement</i>	
•	<i>Abstract</i>	
•	<i>List of Tables</i>	
•	<i>List of Charts</i>	
•	<i>Definitions of Notation</i>	
1.	INTRODUCTION	1
2	LITERATURE REVIEW	5
3.	PROBLEM FORMULATION & SOLUTION METHOD	
3.1	Problem Formulation	15
3.2	Solution Method	19
4.	COMPUTATIONAL FORMULATION AND ANALYSIS	
4.1	Introduction	32
4.2	Small Size Transshipment Problems	35
4.3	Medium Size Transshipment Problems	40
4.4	Large Size Transshipment Problems	42
4.5	Remark on no. of Iterations taken by our Heuristic	49
4.6	Running the Heuristic in $O(c n^2)$ times.	50
5.	CONCLUSIONS AND DIRECTIONS FOR FUTURE RESEARCH	57
	REFERENCES	58
	APPENDICES .	
	Appendix I : Solved Numerical Examples.	61
	Appendix II : Pascal Program Listing .	71
	CHARTS :	
	• Charts showing percentage closeness to the optimal result.	
	• Charts showing comparison between no. of iterations taken by the heuristic procedure and CPLEX.	

LIST OF TABLES

Table No.	Problem Size	Description	Number of Problems Tested	Page No
4.1	3 x 3 x 3	Small Size Problems	15	36
4.2	4 x 4 x 4	Small Size Problems	15	37
4.3	5 x 5 x 5	Small Size Problems	10	38
4.4	7 x 7 x 7	Small Size Problems	10	39
4.5	10 x 10 x 10 to 75 x 75 x 75	Medium Size Problems	10	41
4.6	100 x 100 x 100	Large Size Problems	50	43
4.7	150 x 150 x 150	Large Size Problems	50	45
4.8	200 x 200 x 200	Large Size Problems	50	47
4.9	100 x 100 x 100	Results from $O(c n^2)$ Solution Procedure	50	51
4.10	150 x 150 x 150	Results from $O(c n^2)$ Solution Procedure	50	53
4.11	200 x 200 x 200	Results from $O(c n^2)$ Solution Procedure	50	55

LIST OF CHARTS

Charts Showing Percentage Closeness

Problem Size	Page No.
3 x 3 x 3	83
4 x 4 x 4	83
5 x 5 x 5	84
7 x 7 x 7	84
100 x 100 x 100	85
150 x 150 x 150	86
200 x 200 x 200	87

Charts Showing Comparisons of Iterations Taken by Heuristic and CPLEX

Problem Size	Page No.
3 x 3 x 3	88
4 x 4 x 4	88
5 x 5 x 5	89
7 x 7 x 7	89
100 x 100 x 100	90
150 x 150 x 150	91
200 x 200 x 200	92

DEFINITIONS OF NOTATION

Notation	Definition
I	Number of Plants
J	Number of Intermediate Points/Warehouses
K	Number of Markets
$X_{1,j}$	The Quantity Recieved as a Fraction of Total Market Demand from Plant i to Intermediate Point j
$X_{2,j,k}$	The Quantity Recieved as a Fraction of Total Market Demand from Intermediate Point j to Market k
D_k	The Absolute Value of Demand at Market k
d_k	The Demand at Market k as a Fraction of Total Market Demand
$C_{1,j}$	Cost of Shipping $\sum_{k=1}^K D_k$ units from Plant i to Intermediate Point j
$C_{2,j,k}$	Cost of Shipping $\sum_{k=1}^K D_k$ units from Intermediate Point j to Market k
B_i	The Absolute Value of the Plant Capacity at Plant Location i
b_i	The Capacity of the Plant i as a Fraction of total Market Demand
$C_{1,i}^*$	$\min_j (C_{1,i,j}) \quad \forall i = 1..I$
$C_{2,k}^*$	$\min_j (C_{2,j,k}) \quad \forall k = 1..K$
m_i	$\{j : C_{1,i}^* = C_{1,i,j}\}$
l_k	$\{j : C_{2,k}^* = C_{2,j,k}\}$
WJ	Set of W_j 's that can be Increased
$C_{22,k}$	$\min_{j \in \{1..J\}-WJ} C_{2,j,k} \quad \forall k$
$V_{1_0}, z_1, V_{2_0}, v_k, W_1$	The Dual Variables Associated with Constraints (2),(3),(4),(5),(6) respectively .
n	The Sum of Supply, Intermediate and Demand Points

CHAPTER 1

INTRODUCTION

The uncapacitated transshipment problem is a special case of minimum cost flow problem which includes all the general features of minimum cost flow problem except for not having (finite) arc capacities . In Transshipment problem , flow takes place in stages, from plant goods move to warehouses and from warehouses they move to the markets.

The Transshipment problem is an example of linear network optimization and is now a standard application for industrial firms having several manufacturing plants , warehouses , sales territories and distribution outlets . In this instance , the strategic decisions involve selecting transportation routes so as to allocate the production of various plants to several warehouses or terminal points.

A large class of network flow problems centre around the shipping and distribution problems . This problem is best described in terms of shipments from plants to warehouses and from warehouses to the markets .

Network flow is an important area in linear programming and is known for its novel theoretical properties . Most network flow problems are linear programming problems and their special structure often allow us to develop highly efficient algorithms that do not seem possible for the general linear programming problems . This efficiency in solving network flow problems has motivated the practitioners to identify the class of problems that can be modelled as network flow problems . Many successes have been made at this front and a wide variety of practical problems not only in transportation and

communication , but also in facility location , production planning , project management , machine loading , operation scheduling and cash management can be formulated and solved as network flow problems . Recent developments in solution methodology , implementation technology and improved computer software have enabled the network flow problems to be handled much more effectively and conveniently .

Network flow problems are found in various types . Some well known types of network flow problem are :

- 1 The Shortest Path Problem
2. The Maximum Flow Problem
- 3 The Transportation Problem
4. The Transshipment Problem
5. The Assignment Problem
6. The Minimum Cost Flow Problem

In the above mentioned types of network flow problems , the first five types (i.e 1-5) are special cases of the minimum cost flow problems and can be solved even more efficiently than the minimum cost flow problems . The minimum cost flow problems is however the most popular network flow problem . Some of the reason for its popularity include

- Simplicity and intuitive appeal of the problem as well as its solution procedures ;
- Integer valued optimum solution ;
- Little computer memory requirement ;
- Efficiency of computer codes in solving even the large sized problems ;
- The ability to formulate several complex problems as a sequence of the minimum cost flow problems .

Due to these reasons , the minimum cost flow problem has been extensively researched in the literature . Researchers have developed numerous algorithms . In the past , polynomial algorithms for the minimum cost flow problems have been vigorously pursued and several new algorithms have been developed . Among these contributions are the cost scaling algorithm due to Goldberg and Tarjan [1987] and the double scaling algorithm due to Ahuja, Goldberg, Orlin and Tarjan [1988] . These algorithms have very attractive worst case time bounds but their empirical performance is unclear .

In most of the above mentioned cases , the well known primal approaches solve the uncapacitated transshipment problem in $O(n^3 \log(n))$ times . In this paper we tried to give a dual based procedure which takes $O(c n^3)$ times to solve the transshipment problem , where n is the number of nodes in the network . Though the solution given by us in this paper does not give an optimal solution, but covers most of its path towards optimality in $O(c n^3)$ time and thereafter the conventional dual based algorithms can take over to reach the optimal solution

The organization of the thesis is as follows :

In chapter two , we have given relevant literature review which highlight the various literature available in the field of the minimum cost flow problems inclusive of the best available algorithm at present . The problem formulation and the heuristic solution procedure to solve the transshipment problem is given in chapter three . In the following chapter four, we have attempted to provide our computational experience and analysis obtained during the testing of the heuristic on small as well as large sized randomly generated problems . In chapter five we wind up by giving conclusion and directions for future research in this field . Appendix I contains few illustrated numerical examples to

provide illustration of the solution procedure given by us in this thesis . Appendix II contains the computer program developed by us to carry out the step by step heuristic solution procedure . We also attached various graphs to illustrate our computationally efficient $O(cn^3)$ procedure

Our solutions are within 98 % of the optimal solution . Our solution can be used as an advanced start by the conventional dual based approaches for the transshipment problem . One more interesting feature of our thesis is that an $O(cn^2)$ procedure developed by us took us to 96.5 % closer to the optimal solution .

CHAPTER 2

LITERATURE REVIEW

The minimum cost flow problem is an important network flow problem . The transshipment problem , a special case of minimum cost flow problem , was posed and solved by A. Orden [1956] , Koopmans [1947] and Kantorovich [1939] .

Minty [1960] and Fulkerson [1961] independently discovered the out-of-kilter algorithm . Aashtiani and Magnanti [1976] have described an efficient implementation of this algorithm . The cycle-canceling algorithm is credited to Klien[1967] . Three special implementations of the cycle-canceling algorithm run in polynomial time : the first , due to Barahona and Tardos [1989] (which , in turn , modifies an algorithm by Weintraub [1974]) , augments flow along (negative) cycles with the maximum possible improvements ; the second , due to Goldberg and Tarjan[1988] , augments flow along minimum mean cost (negative) ; the third , due to Wallachar and Zimmerman [1991] , augments flow along minimum ratio cycles .

Helgason and Kennington [1977] and Armstrong , Klingman and Whitman[1980] describe the specialization of the linear programming dual simplex algorithm for the minimum cost flow problem . Each of these algorithms perform iterations that can (apparently) not be polynomially bounded. Zadeh [1973 a] describes one such example on which each of several algorithms - the primal simplex algorithm with Dantzig's pivot rule , the dual simplex algorithm , the negative cycle algorithm , the successive shortest path algorithm , the primal dual algorithm , and the out-of-kilter algorithm - performs an exponential number of iterations . Zadeh [1973 b] has also described more pathological examples for network algorithms .

The network simplex algorithm and its practical implementations have been most popular with operations researchers . Johnson [1966] suggested the first tree manipulating data structure for implementing the simplex algorithm . The first implementations using the ideas , due to Srinivasan and Thompson [1973] , and Glover , Klingman and Napier [1974], significantly reduced the running time of the simplex algorithm Glover, Klingman and Stutz [1974], Bradley, Brown and Graves [1977] and Barr, Glover and Klingman [1979] subsequently discovered improved data structures . The book of Kennington and Helgason [1980] is an excellent source for references and background material concerning these developments .

The relaxation algorithms proposed by Bertsekas and his associates are other attractive algorithms for solving the minimum cost flow problems and its generalization . For the minimum cost flow problem , this algorithm maintains a pseudo flow satisfying the optimality conditions The algorithm proceeds by either

- augmenting flow from an excess node to a deficit node along a path consisting of arcs with reduced cost, or
- changing the potentials of subset of nodes .

In the later case , it resets flows on some arcs to their lower or upper bounds so as to satisfy the optimality conditions ; however , this flow assignment might change the excesses and deficits at nodes . The algorithm operates so that each change in the node potentials increases the dual objective function value and when it finally determines the optimal dual objective function value , it has also obtained the optimal primal solution . This relaxation algorithm has exhibited nice empirical behaviour . Bertsekas [1985] suggested the relaxation algorithm for the minimum cost flow problem (with integer data) . Bertsekas and Tseng [1985] extended this approach for the

minimum cost flow problem with real data and the generalized minimum cost flow problem .

Ahuja , Batra , Gupta and Punnen [1996] address the problem of allocating a given budget to increase the capacities of arcs in a transshipment network to minimum cost flow in the network. the capacity expansion costs of arcs are assumed to be piece wise linear convex functions . They used the properties of the optimum solution to convert this problem into a parametric network flow problem . The concept of optimum basis structure is used which allows to consider piece wise linear convex functions without introducing additional arcs . The resulting algorithm yields an optimum solution of the capacity expansion problem for all budget levels less than or equal to the given budget . For integer data this algorithm performs almost all computations in integer.

A number of empirical studies have extensively tested minimum cost flow algorithms for wide variety of network structures , data distributions , and problem size. The most common problem generator is NETGEN , due to Klingman , Napier and Stutz[1974] , which is capable of generating assignment, capacitated or uncapacitated transportation / transshipment and minimum cost flow problems Glover , Karney and Klingman [1974] and Aashtiani and Magnanti [1976] have tested the primal dual and out-of-kilter algorithms . Helgason and Kennington [1977] and Armstrong , Klingman and Whitman [1980] have reported on extensive studies of the dual simplex algorithm . The primal simplex algorithm has been a subject of more rigorous investigation ; studies conducted by Glover , Karney , Klingman and Napier [1977] , Glover, Karney and Klingman [1974] , Bradley , Brown and Graves [1977] , Mulvey [1978 b] , Grigoriadis and Hsu [1979] and Grigoriadis [1986] are noteworthy . Bertsekas and Tseng [1988] have presented computational results for the relaxation algorithm .

Bertsekas and Tseng [1988 b] developed the relaxation algorithm and conducted extensive computational investigations of it . A FORTRAN code for the relaxation algorithm appears in Bertsekas and Tseng [1988 a] . Their study and those conducted by Grigoriadis [1986] and Kennington and Wang [1990] indicate that the relaxation algorithm and the network simplex algorithm are the two fastest available algorithms for solving the minimum cost flow problem in practice . When the supplies/demands at nodes are relatively small , the successive shortest path algorithm is the fastest algorithm . Previous computational studies conducted by Glover , Karney and Klingman [1974] and Bradley , Brown and Graves [1977] have indicate that the network simplex algorithm is consistently superior to the primal-dual and out-of-kilter algorithms.

Polynomial Time Algorithm

In the recent past , researchers have actively pursued the design of fast (weakly) polynomial and strong polynomial time algorithms for the minimum cost flow problems . Recall that a algorithm is strongly polynomial-time if its running time is polynomial in the number of nodes and arcs , and does not involve terms containing logarithms of C or U . The table given below summarizes these theoretical developments in solving the minimum cost flow problem . The table also reports running time for networks with ' n ' nodes and ' m ' arcs , m' of which are capacitated . It assumes the integral cost coefficients are bounded in absolute value by U . the terms $S(.)$ is the running time for the shortest path problem and the term $M(.)$ represents the corresponding running time to solve a maximum flow problem.

S. No.	Discoverers	Year	Running time
1	Edmonds and Karp	1972	$O((n+m) \log U S(n, m, nC))$
2	Rock	1980	$O((n+m) \log U S(n, m, nC))$
3	Rock	1980	$O((n \log C M(n, m, U)))$
4	Bland and Jensen	1985	$O((m \log C M(n, m, U)))$
5	Goldberg and Tarjan	1987	$O((nm \log (n^2/m) \log (nC)))$
6	Goldberg and Tarjan	1988	$O((nm \log (n) \log (nC)))$
7	Ahuja , Goldberg, Orlin and Tarjan	1988	$O((nm \log(\log U) \log (nC)))$

Strongly Polynomial-Time Algorithms

S. No.	Discoverers	Year	Running Time
1	Tardos	1985	$O(m^4)$
2	Orlin	1984	$O((n+m)^2 \log n S(n,m))$
3	Fujishige	1986	$O((n+m)^2 \log n S(n,m))$
4	Galil and Tardos	1986	$O((n^2 \log n S(n,m)))$
5	Goldberg and Tarjan	1987	$O((nm^2 \log n \log (n^2/m)))$
6	Goldberg and Tarjan	1988	$O(nm^2 \log^2 n)$
7	Orlin	1988	$O((n+m) \log n S(n,m))$

For the sake of comparing the polynomial and strongly polynomial-time algorithms, we make the assumption the $C = O(n^k)$ and $U = O(n^k)$ for the same constant 'k'. Under this assumption the best bounds for the shortest path and maximum flow problem are :

Polynomial Time Bounds	Discoverers
$S(n,m,c) = O(\text{Min}(m+n\sqrt{\log C}), (m \log \log C))$	Ahuja, Mehlhorn, Orlin and Tarjan [1990]
$S(n,m,c) = O(\text{Min}(m+n\sqrt{\log C}), (m \log \log C))$	Van Emde Baas, Kaas and Zijstra [1977]
$M(n,m,U) = O(nm \log(n/m(\sqrt{\log U})+2))$	Ahuja, Orlin, and Tarjan [1989]

Strongly polynomial-Time Bounds :

$S(n,m) = O(m+n \log n)$	Fredman and Tarjan [1984]
$M(n,m)=O(\min(mn+n^{2+\varepsilon}, nm \log n))$	King, Rao, Tarjan [1991]

where ε is any fixed constant.

Now we give some well known scaling algorithms used in the study of minimum cost flow problems along with their running times :

Algorithm	Running Time
Capacity Scaling Algorithm	$O((m \log U)(m+n \log n))$
Cost Scaling Algorithm	$O(n^3 \log (nC))$
Double Scaling Algorithm	$O(nm \log U \log (nC))$
Minimum Mean Cycle-Canceling Algorithm	$O(n^2 m^3 \log n)$
Repeated Capacity Scaling Algorithm	$O((m^2 \log n)(m+n \log n))$
Enhanced Capacity Scaling Algorithm	$O((m \log n)(m+ n \log n))$

Using capacity and right-hand-side scaling, Edmonds and Karp[1972] developed the first (weakly) polynomial - time algorithm for the minimum cost flow problem , known as the RHS-scaling algorithm for the minimum cost flow problem . A variant of the Edmonds-Karp algorithm , was suggested by Orlin [1988] . the scaling technique did not initially capture the interest of many researchers , since they regarded as having little practical utility. However , researchers gradually recognized that the scaling technique has great theoretical value as well as potential practical significance

Rock[1980] developed two different bit-scaling algorithms for the minimum cost flow problems, one using scaling and the other using cost scaling. This cost scaling algorithm reduces the minimum cost flow problem to a sequence of $O(n \log C)$ maximum flow problems. Bland and Jensen [1985] independently discovered a similar cost scaling algorithm.

The pseudo flow push algorithms for the minimum cost flow problem use the concept of approximate optimality, introduced independently by Bertsekas [1979] and Tardos [1985]. Bertsekas [1986] developed the first pseudo flow push algorithm. This algorithm was pseudopolynomial-time. Goldberg and Tarjan [1979] used a scaling technique on a variant of this algorithm to obtain a generic pseudo flow push algorithm. Tarjan[1984] proposed a wave algorithm for the maximum flow problem. the wave algorithm for the minimum cost flow problem was developed independently by Goldberg and Tarjan [1987] and Bertsekas and Eckstein[1988], relies upon the similar ideas. Using a dynamic tree data structure in generic pseudo flow push algorithm, Goldberg and Tarjan [1987] obtained a computational time bound of $O(nm \log n \log(nC))$. They also showed that the minimum cost flow problem can be solved using $O(n \log(nC))$ blocking flow computations. Using both finger tree and dynamic tree data structures, Goldberg and Tarjan [1982] obtained a $O(nm \log(n^2/m) \log(nC))$ bound for the wave algorithm. this algorithm was subsequently improved to $O(nm \log U \log(nC))$ by Ahuja, Goldberg, Orlin and Tarjan[1982].

The first strongly polynomial-time minimum cost flow problem is due to Tardos[1985]. Several researchers including Orlin[1984], Fujishige[1986], Gail and Tardos[1986] and Orlin[1988] provided subsequent improvements in the running time. Goldberg and Tarjan [1988 a] obtained another strongly polynomial-time algorithm by

slightly modifying their pseudo flow push algorithm. Goldberg and Tarjan [1988 b] also show that their algorithm that proceeds by canceling minimum mean cycles is strongly polynomial time. Currently, the fastest strongly polynomial-time algorithm is due to Orlin[1988]

Some additional polynomial-time minimum cost flow algorithms include (1) a triple scaling algorithm due to Gabow and Tarjan [1989 a], (2) a special implementation of the cycle canceling algorithm developed by Garahona and Tardos[1989], and (3) (its dual approach) a cut canceling algorithm proposed by Ervolina and McCormick [1990 a] .

Interior point linear programming algorithms are another source of polynomial-time algorithms for the minimum cost flow problem. Among these, the fastest available algorithm, due to Vaidya[1989] , solves the minimum cost flow problem in $O(n^{2.5} \sqrt{mK})$ time, with $K = \log n + \log C + \log U$.

Currently the best available time bound for the minimum cost flow problem is $O(\min \{ nm \log (n^2/m) \log (nC) \} , nm (\log \log U) \log (nC), (m \log n)(m+n \log n) \})$; the three bounds in this expression are, respectively, due to Goldberg and Tarjan[1987], Ahuja, Goldberg, Orlin and Tarjan [1992] and Orlin[1988] . At this time the research community has yet to develop sufficient evidence to fully assess the computational worth of scaling and interior point linear programming algorithms for the minimum cost flow problem

Ali, Padman and Thiagarajan [1989] reported that the number of pivots in implementations of the dual algorithm is smaller than in the primal simplex algorithm. This feature has obscured the true performance characteristics of the dual algorithms. Sharma

and Sharma [1998] also gave a dual based procedure to solve the simple transportation problem in $O(c n^2)$ time .

In the present work we also tried to extend the dual feature as given by Sharma and Sharma[1998] to the transshipment problem. We , in this thesis , have developed a heuristic solution procedure for solving the transshipment problem in $O(c n^3)$ time, the details of which is given in Chapter three .

CHAPTER 3

PROBLEM FORMULATION AND SOLUTION METHOD

In this chapter we give the problem to be solved in this paper and the dual based solution procedure to solve the problem We start by defining the constants used by us in the problem and subsequently we have given the complete problem formulation and the dual based solution procedure

3.1 PROBLEM FORMULATION

3.1.1 Constants of the Problem :

In this paper we used the index i for supply nodes , index k for demand nodes and index j for the intermediate / transshipment nodes .

D_k is the demand at demand node k and d_k is the demand at demand node k as a fraction of total demand Hence,

$$d_k = \frac{D_k}{\sum_{k=1}^K D_k}$$

and obviously we have,

$$\sum_{k=1}^K d_k = 1$$

B_i is the supply available at source node i and b_i is the supply available as a fraction of total market demand i.e. ,

$$b_i = \frac{B_i}{\sum_{k=1}^K D_k}$$

and we assume that ,

$$\sum_{i=1}^I b_i = 1$$

$C1_{i,j}$ is the transportation cost from source node i to intermediate node j for transporting the entire market demand $\sum_{k=1}^K D_k$

$C2_{j,k}$ is the transportation cost from intermediate node j to demand node k for transporting the entire market demand $\sum_{k=1}^K D_k$

[In this paper we assume that total supply from all the sources is equal to the total demand of all the demand nodes .]

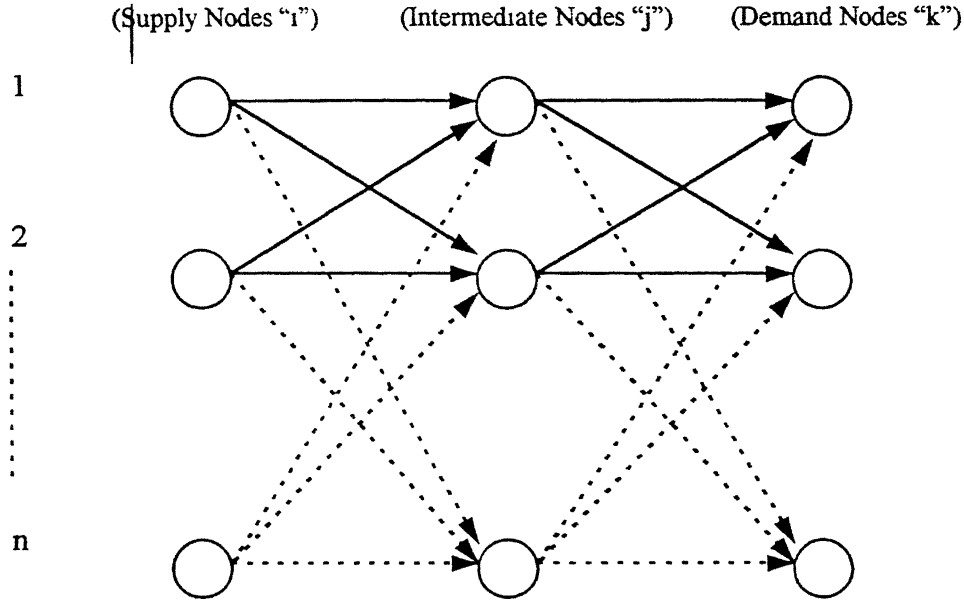
3.1.2 Decision Variables :

$X1_{i,j}$ is the quantity received as a fraction of total demand at intermediate node j from source node i

$X2_{j,k}$ is the quantity received as a fraction of total demand at demand node k from intermediate node j .

3.1.3 Problem Formulation :

We state the problem to be solved in this paper as follows :



Problem P :

$$\text{Minimize } Z = \sum_{(i,j)} C1_{ij} \cdot X1_{ij} + \sum_{(j,k)} C2_{jk} \cdot X2_{jk} \quad (1)$$

Subject to

$$\sum_{(i,j)} X1_{ij} = 1 \quad \forall i, j \quad (2)$$

$$-\sum_{j=1}^J X1_{ij} \geq -b_i \quad \forall i \quad (3)$$

$$\sum_{(j,k)} X2_{jk} = 1 \quad \forall j, k \quad (4)$$

$$-\sum_{j=1}^J X2_{jk} \geq -d_k \quad \forall k \quad (5)$$

$$\sum_{i=1}^I X1_{ij} - \sum_{k=1}^K X2_{jk} = 0 \quad (6)$$

$$X_{1ij} \geq 0 \quad \forall i, j \quad (7)$$

$$X_{2jk} \geq 0 \quad \forall j, k \quad (8)$$

In the model defined above , we assumed that the total supply at all supply nodes is equal to the total demand at all the demand nodes ,which makes it a valid formulation of the transshipment problem

3.1.4 Dual of the Problem P :

Now we define the dual of the problem P defined above . In the dual of the problem , we associate the dual variables V_{10} , z_i , V_{20} , v_k , and W_j with constraints (2) , (3) , (4) , (5) and (6) respectively .

The dual of the problem model P (1) - (8) is as follows .

Problem DP :

$$\text{Maximize } Z_{\text{dual}} = (V_{10} - \sum_{i=1}^I z_i \cdot b_i) + (V_{20} - \sum_{k=1}^K d_k \cdot v_k) \quad (9)$$

Subject to :

$$V_{10} - z_i \leq C_{1ij} - W_j \quad \forall i, j \quad (10)$$

$$V_{20} - v_k \leq C_{2jk} + W_j \quad \forall j, k \quad (11)$$

$$z_i \geq 0 \quad \forall i \quad (12)$$

$$v_k \geq 0 \quad \forall k \quad (13)$$

$$V_{10}, V_{20}, W_j \quad \forall j \quad \text{unrestricted.} \quad (14)$$

3.2 SOLUTION PROCEDURE

The solution procedure is based on the dual of the problem P i.e. the problem DP and it aims to increase the value of Z_{dual} as much as possible . We start by defining a special case of the problem i.e. the case when all W_j 's are zero's

3.2.1 Special Case of the Problem

In this special case of the problem DP i.e. when all W_j 's are zero the problem DP has the following form .

Problem DP1 :

$$\text{Maximize } (V1_0 - \sum_{i=1}^I z_i \cdot b_i) + (V2_0 - \sum_{k=1}^K d_k \cdot v_k) \quad (15)$$

Subject to

$$V1_0 - z_i \leq C1_{i,j} \quad \forall i, j \quad (16)$$

$$V2_0 - v_k \leq C2_{j,k} \quad \forall j, k \quad (17)$$

$$z_i \geq 0 \quad \forall i \quad (18)$$

$$v_k \geq 0 \quad \forall k \quad (19)$$

$$V1_0, V2_0 \quad \forall j \quad \text{unrestricted.} \quad (20)$$

Problem DP1 separates into two independent problems which are identical to problem DP2 given below -

Problem DP2

$$\text{Maximize } V_0 - \sum_{k=1}^K d_k \cdot v_k \quad (21)$$

Subject to,

$$V_0 - v_k \leq C_{i,k} \quad \forall i,k \quad (22)$$

$$v_k \geq 0 \quad (23)$$

$$V_0 \quad \text{unrestricted} \quad (24)$$

The two independent problems are DP11 and DP12 which are given below

Problem DP11 .

$$\text{Maximize} \quad (V1_0 - \sum_{i=1}^I z_i \cdot b_i)$$

Subject to,

$$(16),(18) \text{ and } V1_0 \quad \text{unrestricted}$$

Problem DP12

$$\text{Maximize} \quad (V2_0 - \sum_{k=1}^K d_k \cdot v_k)$$

Subject to,

$$(17) , (19) \text{ and } V2_0 \quad \text{unrestricted}$$

Sharma[26] has given an efficient algorithm to optimally solve the problem

DP2 and it is reproduced here for the sake of completeness

3.2.1.1 Algorithm for solving the Problem DP2 :

Step 1 Find $C_k^* = \min_i (C_{ik})$ for all $k = 1, K$ and remove all the redundant constraints (of the equation (22)) in problem DP2

In case of a tie , only one equation is retained and all the others are eliminated. Then the problem DP2 becomes the following

Problem DP21

$$\text{Maximize} \quad (V_0 - \sum_{k=1}^K d_k \cdot v_k) \quad (25)$$

Subject to

$$V_0 - v_k \leq C_k^* \quad \forall k \quad (26)$$

$$v_k \geq 0 \quad \forall k \quad (27)$$

$$V_0 \quad \text{unrestricted} \quad (28)$$

Step 2 : We sort the values of C_k^* in an increasing order and reindex such that $C_r^* \leq C_{r+1}^*$ for all $r = 1..K-1$

Step 3 : Since $\sum_{k=1}^K d_k = 1$

we set, $v_k = V_0 - C_k^*$ for all k .

It has been shown in Sharma[1998] and Sharma[1996] that above algorithm solves the problem DP21 optimally

3.2.2 Definitions :

Now , in this section , we give some of the definitions which are frequently used in the solution procedure .

$$C1_i^* : \min_j (C1_{ij}) \quad \forall i, j$$

$$m_i : \{j \mid C1_i^* = C1_{ij}\} \quad \forall i, j$$

$$C2_k^* : \min_j (C2_{jk}) \quad \forall j, k$$

$$l_k : \{j \mid C2_k^* = C2_{jk}\} \quad \forall j, k$$

WJ : Set of W_j 's to increase the value of the objective value Z_{dual}

$$C22_k : \min_{j \in \{1, \dots, J\} - WJ} C2_{jk} \quad \forall k$$

3.2.3 Objective :

We want to increase a collection of W_j s . $j \in WJ$ so that objective value (Z_{dual}) of the problem DP [(9)-(14)] increases

If we increase W_j 's then $C1_i^*$ for the problem DP11 decrease and objective function value of problem DP11 decreases but at the same time $C2_k^*$ for the problem DP12 increases and hence objective function value of problem DP12 increases. We have to increase W_j 's . $j \in WJ$ in such a manner as to strike the balance These are the ideas of all the heuristics presented below

3.2.4 Heuristic H 1 :

In this section we give the heuristic which calculates the net benefit obtained by unit increase of the variables W_j 's . $j \in WJ$. The execution of this heuristic tells whether it will be beneficial to increase the combination of W_j 's . $j \in WJ$ or not . Only if the net benefit (as calculated using this heuristic) is greater than zero, we increase the currently obtained combination of W_j 's . $j \in WJ$ otherwise we go to find the next combination of W_j 's : $j \in WJ$ to be increased The Heuristic is defined as follows :

Given a set of WJ , we determine if it is beneficial to increase W_j ' s : $j \in WJ$ so that the objective value of the problem DP [(9)-(14)] increases

Step 1 : Set benefit =0.

Compute $C1_i^*$, m_i $\forall i$ [as defined above]

Compute $C2_k^*$, l_k $\forall k$ [as defined above]

Step 2 :

for $k = 1$ to K do

if $I_k \subseteq WJ$ then

Set benefit = benefit + d_k

Step 3 Set loss = 0

Step 4 for $j = 1$ to J

if ($j \in m_i$) and ($j \in WJ$) then

set loss = loss + b_i

[Note : b_i 's to be considered only once for each i .]

Step 5 Compute Net_Benefit = benefit - loss

If (Net_Benefit > 0) then

It is possible to improve the objective functional value of the problem DP by increasing the value of W_j s. $j \in WJ$ by at least one unit (We assume that all costs are integer.)

Result 1 : The complexity of the heuristic H1 is $O(n)$ times

Proof : In the heuristic H1 defined above the steps 1 is executed in $O(I+K)$ steps while step 2 is done in $O(K)$ steps and step 4 is done in $O(J)$ steps . Thus the Heuristic H1 runs in $O(n)$ number of steps where $n = I + J + K$, i.e. the sum of supply, demand and intermediate points

Now we give two heuristics below to determine the set of WJ (as defined above).

3.2.5 Heuristic H2 :

Using this heuristic , we find the combination of W_j 's $j \in WJ$ which is required to increase the value so that the value of the objective function value Z_{dual} can be improved. In this heuristic , we calculate the combination of W_j 's $j \in WJ$ sequentially i.e. taking each l_k at a time as the set WJ . The detailed heuristic is given below in easy steps

Step 1 : Compute $C1_i^*$, m_i $\forall i$ [as defined above]
 Compute $C2_k^*$, l_k , $C22_k$ $\forall k$ [as defined above]

Step 2 : for $k = 1$ to K do

 Begin

 Set $WJ = l_k$

 Compute Net_Benefit by using Heuristic H1

 If (Net_Benefit > 0) then

 Stop. We have found a collection of W_j 's $j \in WJ$

 which can be increase in value to improve the

 objective function value of the problem DP.

 Stop.

 End.

Result 2 : The Heuristic H2 runs in $O(n^2)$ number of steps

Proof : In the heuristic H2 above , step 1 is executed at the most $O(I+K)$ steps and step 2 is executed at the most $O(K^2)$ times. Hence heuristic H2 is executed for at the most $O(n^2)$ number of steps where $n = I+J+K$

Now we define another heuristic for finding the set WJ

3.2.6 Heuristic H3 :

Now in this section we give one more heuristic which can be used to find the combination of W_j 's : $j \in WJ$ to improve the objective function value of Z_{dual} . In this heuristic, we find the combination of W_j 's : $j \in WJ$ in a different way. Here we take one l_k and find out the common elements of this particular l_k with all others l_k 's. After this we include all such common element in the set WJ. For a better understanding of this heuristic we give below the complete heuristic in steps below :

Step 1 . Compute $C1_i^*$, m_i $\forall i$ [as defined above]
 Compute $C2_k^*$, l_k , $C22_k$ $\forall k$ [as defined above]

Step 2 . For $k1 = 1$ to K do

 Begin

$WJ = l_{k1}$

 For $k2 = (k1+1)$ to K do

 If l_{k1} and l_{k2} both have some common elements then

 Set $WJ = WJ + l_{k2}$

 Compute Net_Benefit by using Heuristic H1

 If (Net_Benefit > 0) then

 Stop We have found a collection of W_j 's : $j \in WJ$

 which can be increase in value to improve the
 objective function value of the problem DP.

 Stop

End

Result 3 : The heuristic H3 runs in $O(n^3)$ number of steps .

Proof : In the above stated Heuristic H3, step 1 is executed for at the most $O(I+K)$ number of steps whereas the step 2 executes for at the most $O(K1*K2*O(n))$ number of steps Hence the complete heuristic H3 runs in $O(n^3)$ times where $n = I+J+K$

After defining the heuristics for determining the set WJ and calculating the net benefit which enable us as to which particular combination of W_j 's $j \in WJ$ are to be increased finally we require upto how many units we can really increase the values of the W_j 's $j \in WJ$ to improve the objective function value of Z_{dual} . This task is given by us in heuristic H4 defined in the coming section

3.2.7 Heuristic H4 :

This heuristic calculates the extent of increase of the combination of W_j 's $j \in WJ$ to improve the value of objective function Z_{dual} . To calculate this, we first find out that upto which extent we can increase the values of W_j 's $j \in WJ$. For this purpose we find extent of increase for each k as the difference between the smallest and the second smallest value of $C2_{jk}$ of that particular k . After this, we take the minimum of all such values obtained We also calculate that upto which extent we can decrease the values of W_j 's $j \in WJ$ as it also effects the improvement of the solution To find the extent of decrease , we first find out the extent of decrease for each value of matrix $C1$ such that $j \in WJ$. Then we find for each i , the extent of decrease possible in a specified

manner [given below] and take the minimum of all such values to finally obtain the value upto which we can decrease the values of W_j , $s : j \in WJ$. Now after calculating the value of possible increase and possible decrease, we finally take the minimum of the two which gives us finally the value with which we can update the cost coefficients. This whole procedure is described below in detailed steps

Step 1 : Given a set of WJ .

$$\text{Set Extent_of_Increase}_k = \infty \quad \forall k$$

Step 2 : Set $\text{Extent_of_Increase}_k = C22_k - C2_k^*$

Step 3 : $\text{Extent_of_Increase2} = \min_k [\text{Extent_of_Increase}_k]$

Step 4 : $\text{Extent_of_Decrease}_{ij} \{ j \in WJ \}$

$$= C1_{ij} - C1_i^* \quad \text{if } C1_{ij} - C1_i^* > 0$$

$$= \infty \quad \text{if } C1_{ij} - C1_i^* = 0$$

$$\text{Extent_of_Decrease}_i$$

$$= \min \{ \text{Extent_of_Decrease}_{ij} \} \quad \forall \text{Extent_of_Decrease}_{ij} \neq \infty$$

$$= \infty \quad \text{If } \exists \text{Extent_of_Decrease}_{ij} = \infty$$

Step 5 : $\text{Extent_of_Decrease1} = \min_i \{ \text{Extent_of_Decrease}_i \}$

Step 6 : $\text{Extent_of_Increase} = \min \{ \text{Extent_of_Increase2}, \text{Extent_of_Decrease1} \}$

Stop

Result 4 : The Heuristic H4 runs in $O(n^2)$ times .

Proof : In the above defined heuristic, Steps 1,2,3 runs in $O(K)$ number of steps
Step 5 runs in $O(I)$ number of steps .Step 4 is executed for $O(I*J)$
times. Hence the computational complexity of the heuristic H4 is $O(n^2)$
times where $n = I+J+K$.

Finally we give a complete Heuristic H5 which we used to increase the
objective function value of the problem DP using the heuristics H1 - H4.

3.2.8 Heuristic H5 :

In this section we describe the complete solution procedure which uses all
the above defined heuristic to obtain the best solution . This whole procedure is given in
the following lines in very easy steps

Step 1 : Compute $C1_i^*$, m_i $\forall i$ [as defined above]

Compute $C2_k^*$, l_k $\forall k$ [as defined above]

Step 2 Find set WJ using heuristic H2

Step 3 : Compute Net_Benefit for the set WJ using H1.

If (Net_Benefit > 0) then

Go to Step 6

Otherwise Go to Step 4.

Step 4 : Find set WJ using Heuristic H3.

Step 5 : Compute Net_Benefit for the set WJ using H1

If (Net_Benefit > 0) then

Go to Step 6

Otherwise Go to Step 9

Step 6 . Calculate Extent_of_Increase using heuristic H4.

step 7 . Update costs coefficients $C1_{ij}$ and $C2_{jk}$ as follows :

$$C1_{ij} = C1_{ij} - \text{Extent_Of_Increase.} \quad \forall j \in WJ$$

$$C2_{jk} = C2_{jk} + \text{Extent_Of_Increase.} \quad \forall j \in WJ$$

Step 8 : Go to Step 1

Step 9 : Stop. Print the best solution obtained so far.

Thus with the heuristic H5 , we found the best solution

Result 5 : Heuristic H5 runs in $O(c n^3)$ number of steps .

Proof : In heuristic H5 defined above, step (1) runs in $O(I+K)$ number of steps
Step (2) [H2] runs in $O(n^2)$ times Step (4) [H3] runs in $O(n^3)$ number of
steps. Step (3) and (5) [H1] is done in $O(n)$ times. Step (6)[H4] runs in
 $O(n^2)$ number of steps Step (7) runs in $O(I*J)$ number of times and hence
the whole heuristic runs in $O(n^3)$ number of steps

No. of iterations : At each step of the heuristic we are assured of a
strictly positive improvement in values of W_j 's : $j \in WJ$ (that can be raised
in values) and hence solution to problem DP can be improved by a
positive quantity which is proportional to a quantity equal to
 $\min(\delta_1, \delta_2)$ where,

$$\delta_1 = \min_k (C2_{2k} - C2_k^*)$$

$$\delta_2 = \min_i (\text{Extent_of_Decrease}_i) \quad \{ \text{as defined in 3.2.7 [H4]} \}$$

Let this number be denoted by δ . And optimal solution to problem DP is a finite real number bound by sum of all $C1_{ij}$'s and $C2_{jk}$'s. Hence maximum number of iterations required by heuristic H5 is given by $\lceil (\text{sum of all } C1_{ij}\text{'s and } C2_{jk}\text{'s}) / \delta \rceil$ which is equal to c . Thus heuristic H5 runs in $O(c n^3)$ number of times.

CHAPTER 4

COMPUTATIONAL EXPERIENCE AND ANALYSIS

4.1 Introduction

We have tested our heuristic on both small-size as well as large-size transshipment problems and in this chapter we give the computational results obtained by us from testing .

As far as the computational complexity of the heuristic is concerned , we have already proved in the chapter three alongwith the heuristic solution procedure that our heuristic solves the transshipment problem in $O(c n^3)$ steps . Thus it has already been proved that our heuristic requires less computational effort as compared to the other available algorithms to solve the transshipment problem . Therefore in this chapter we put emphasis on the percentage closeness i.e how close is the solution given by our heuristic to the optimal solution and the no. of iterations taken by our heuristic as compared by the linear programming optimization package CPLEX . Here we do not report the CPU time as a measure of our heuristic performance because CPU time depends greatly on finer details of the computational environment and the test problems such as :

- Programming language , compiler , and computer ;
- Implementation style and skill of the programmer ;
- Generators used to generate the random test problems ;
- Particular programming environment .

Because of the multiple sources of variabilities , CPU times are often difficult to replicate , which is contrary to the spirit of scientific investigation . Another

drawback of the use of CPU time is that it is an aggregate measure of empirical performance and does not provide much insight about an algorithm's behaviour . For example , an algorithm generally performs some fundamental operations repeatedly, and a typical CPU time analysis does not help us to identify these 'bottleneck' operations . Identifying the bottleneck operations of an algorithm can provide useful guidelines for where to provide future efforts to understand and subsequently improve an algorithm .

All the transshipment problems we selected for testing are randomly generated .

CPLEX Optimization Package

For comparing the solution given by our heuristic with the optimal solution we use CPLEX Optimization Package available on system 'BHASKAR' . CPLEX can take input in any of the format such as LP format , MPS format and SAV format . We used MPS input format and hence the input format of our heuristic is required to be converted into MPS format (i.e column oriented format) . During the solution process CPLEX reports its progress . The solution process involves two stages . During Phase I, CPLEX searches for the feasible solution . In Phase II, CPLEX searches for the optimal feasible solution . The iteration log periodically displays the current iteration number and either the current scaled infeasibility during Phase I, or the objective function value during Phase II . This information can be useful for monitoring the rate of solution progress . The iteration log display can be modified using the 'SET' command to display differing amounts of data while the problem is being solved . Once the optimal solution has been found , the objective function value , solution time and iteration count are displayed by CPLEX .

Computer Programs

For testing our heuristic on different transportation problems, we prepare three different computer programs . First one is for generating random problems of specified sizes (size ranging from $10 \times 10 \times 10$ to $200 \times 200 \times 200$) . The second one is for converting the input format for our heuristic into MPS format required by CPLEX. And the last one is for performing the step by step heuristic procedure . This last one is the main program of testing the solution procedure given by us in this thesis and it is given in appendix II

Now in the following pages we give the results obtained by us in testing the various size problems as well as their percentage closeness to the optimal solution We start by giving some small sized problems followed by some medium size problems and finally we give the large size problems tested by us

4.2 Small Size Transshipment Problems :

We tested a total of 50 small sized problems. The problem size in this category of problems is less than $10 \times 10 \times 10$ ranging from $3 \times 3 \times 3$ to $7 \times 7 \times 7$. The distribution of the number of problems in this series of various sizes are as follows:

Problem Size	No. of Problems Taken	Costs Range
$3 \times 3 \times 3$	15	$1 - 10^8$
$4 \times 4 \times 4$	15	$1 - 10^8$
$5 \times 5 \times 5$	10	$1 - 10^8$
$7 \times 7 \times 7$	10	$1 - 10^8$

The result obtained by us in testing these small sized problems are given in Table 4.1 - Table 4.4 .

The results coming from these testing show that the solution coming from these small size problems is optimal (in most of the cases) when compared with the results of CPLEX .

TABLE 4.1**PROBLEM SIZE : 3 x 3 x 3**

S. No.	Solution Given by our Heuristic Solution Procedure		Optimal Solution Using CPLEX Optimization Package		Percentage Closeness
	Solution	No. of Iterations	Solution	No. of Iterations	
1	11910.6	6	11911.356	8	99.99
2	27557.5	7	27557.5	10	100
3	16685.1	6	16685.1	9	100
4	22411.5	6	22411.5	10	100
5	20602.2	7	20602.2	10	100
6	22359.9	6	22359.9	9	100
7	25501.2	7	25501.2	8	100
8	23230.4	8	23230.4	9	100
9	23217.2	7	23217.2	10	100
10	23576.1	7	23576.1	8	100
11	15680.5	7	15680.5	9	100
12	16505.9	8	16505.9	8	100
13	12595.6	5	12595.6	8	100
14	27771.9	7	27771.9	8	100
15	21080.3	6	21080.3	9	100

TABLE 4.2**PROBLEM SIZE : 4 x 4 x 4**

S. No.	Solution Given by our Heuristic Solution Procedure		Optimal Solution Using CPLEX Optimization Package		Percentage Closeness
	Solution	No. of Iterations	Solution	No. of Iterations	
1	18105.5	8	18154.2	12	99.73
2	18519.8	8	18525.1	12	99.97
3	18526.0	9	18526.0	11	100
4	16844.5	8	16844.5	11	100
5	13565.3	8	13565.3	11	100
6	15444.2	8	15444.2	14	100
7	18518.4	10	18518.4	13	100
8	82259.6	13	82259.6	11	100
9	10723.7	10	10723.7	12	100
10	10866.9	8	10866.9	11	100
11	15332.4	10	15332.4	12	100
12	83588.6	10	83588.6	11	100
13	11768.6	9	11768.6	13	100
14	12882.0	8	12882.0	11	100
15	17309.5	9	17309.5	14	100

TABLE 4.3**PROBLEM SIZE : 5 x 5 x 5**

S. No.	Solution Given by our Heuristic Solution Procedure		Optimal Solution Using CPLEX Optimization Package		Percentage Closeness
	Solution	No. of Iterations	Solution	No. of Iterations	
1	10096.6	10	10096.6	16	100
2	12176.2	11	12389.0	17	98.28
3	74788.2	15	7478.82	16	100
4	7670.39	10	7733.0	15	99.19
5	10290.6	11	10300.0	14	99.91
6	13214.0	15	13213.9	15	100
7	9401.56	10	9443.0	14	99.56
8	11541.4	11	11541.43	14	100
9	8977.99	11	9118.0	15	98.46
10	12349.6	16	12355.0	16	99.96

TABLE 4.4**PROBLEM SIZE : 7 x 7 x 7**

S. No.	Solution Given by our Heuristic Solution Procedure		Optimal Solution Using CPLEX Optimization Package		Percentage Closeness
	Solution	No. of Iterations	Solution	No. of Iterations	
1	15656.2	19	15656.2	27	100
2	11766.0	21	11766.0	24	100
3	7671.85	14	7723.0	23	99.34
4	8434.69	26	8443.0	25	99.90
5	9247.58	17	9436.0	25	98.00
6	6207.0	19	6225.0	22	99.71
7	10054.4	21	10054.42	20	100
8	7351.84	18	7368.1	21	99.78
9	13356.0	24	13376.0	23	99.85
10	7079.91	20	7079.9	20	100

4.3 Medium Size Transshipment Problems

In this category of transshipment problems , we randomly generate 20 problems with size ranging from $10 \times 10 \times 10$ to $75 \times 75 \times 75$. All these problems are generated using a random generator . The solution of these problems from our heuristic as well as that given by CPLEX are summarized in Table 4.5 .

The problem size distribution in this category of problems is as follows .

Problem Size	No. Of Problems Tested	Costs Range
$10 \times 10 \times 10$	3	$1 - 10^8$
$20 \times 20 \times 20$	3	$1 - 10^8$
$25 \times 25 \times 25$	3	$1 - 10^8$
$40 \times 40 \times 40$	3	$1 - 10^8$
$50 \times 50 \times 50$	3	$1 - 10^8$
$60 \times 60 \times 60$	2	$1 - 10^8$
$75 \times 75 \times 75$	3	$1 - 10^8$

The results coming from these medium sized problems show that for this category of problems our solution is coming within 97-98 % to the optimal solution .

TABLE 4.5**PROBLEM OF SIZES RANGING FROM 10 x 10 x 10 to 75 x 75 x 75**

S. No.	Solution Given by our Heuristic		Optimal Solution Using CPLEX Optimization Package		Percentage Closeness
	Solution	No of Iterations	Solution	No. of Iterations	
1	6685.46	34	6927.67	32	96.50
2	6779.41	36	6961.3	32	97.38
3	5366.81	25	5406.013	29	99.27
4	3931.32	59	4252.513	70	92.44
5	4010.91	74	4080.194	73	98.30
6	4082.95	57	4175.491	69	97.78
7	3341.71	61	3533.188	90	94.58
8	2687.67	105	2765.866	84	97.17
9	2385.34	81	2430.078	83	98.15
10	2101.39	124	2141.465	140	98.12
11	2085.19	100	2139.297	131	97.47
12	1840.84	88	1897.633	149	97.00
13	1470.34	112	1596.834	168	92.07
14	1663.51	186	1699.951	192	97.85
15	1628.35	133	1706.837	175	95.40
16	1423.45	193	1472.38	208	96.67
17	1322.62	321	1390.738	206	95.10
18	1168.33	478	1203.216	269	97.10
19	1001.96	210	1058.416	272	94.66
20	1093.08	169	1139.331	273	95.94

4.4 Large Size Transshipment Problem

In this category of transshipment problems also we generated 150 problem using random generator with size ranging from $100 \times 100 \times 100$ to $200 \times 200 \times 200$. The solution obtained from our heuristic solution procedure as well as that given by CPLEX are summarized in Table 4.6 - Table 4.8. Each table is for a particular size of transshipment problem and contains the summarized details of 50 problems. In each of these table we reported the problem size, Iterations taken by our heuristic and that of CPLEX and percentage closeness to the optimal solution for the solutions obtained using our heuristic solution procedure. The distribution of the various problems tested in this category is as follows :

Problem Size	No. of Problems Taken	Costs Range
$100 \times 100 \times 100$	50	$1 - 10^8$
$150 \times 150 \times 150$	50	10 - 10000
$200 \times 200 \times 200$	50	10 - 10000

The results of the problems as summarized in the following tables show that our heuristic solution procedure for these large size problems gives a solution which is within 98 % (on average) of the optimal.

TABLE 4.6**PROBLEM SIZE : 100 x 100 x 100**

S. No.	No. of Iteration Taken by our Heuristic to give the Solution	No. of Iterations Taken by CPLEX optimization Package to give the Optimal Solution	Percentage Closeness to the Optimal Solution by our Heuristic
1	246	377	96.57959
2	261	380	96.37769
3	333	363	95.20112
4	276	369	97.25020
5	206	375	96.81606
6	315	378	96.25575
7	394	354	96.23152
8	322	385	98.32666
9	276	368	96.01851
10	319	378	96.73304
11	464	377	101.8670
12	350	360	95.65126
13	215	362	97.36515
14	292	364	95.53120
15	435	380	96.29604
16	283	372	96.75471
17	419	391	99.77138
18	310	368	95.52540
19	332	389	94.91795
20	319	374	96.40491
21	297	377	98.17961
22	286	371	96.98930
23	252	389	98.78578
24	381	381	94.69378
25	365	367	94.69325

TABLE 4.6 [Contd ...]

S. No.	No. of Iteration Taken by our Heuristic to give the Solution	No. of Iterations Taken by CPLEX optimization Package to give the Optimal Solution	Percentage Closeness to the Optimal Solution by our Heuristic
26	262	371	96.98930
27	349	384	98.78578
28	336	372	94.69325
29	370	388	97.58082
30	222	386	95.25702
31	213	383	96.39582
32	329	380	95.52060
33	268	375	96.98929
34	283	373	94.69325
35	318	387	96.98982
36	329	382	97.02474
37	310	377	94.69325
38	303	372	95.82753
39	315	395	95.82742
40	282	384	97.58093
41	415	379	98.17963
42	350	374	94.69325
43	336	365	97.58082
44	370	376	95.25645
45	283	363	97.58083
46	291	389	96.40491
47	281	397	97.58083
48	308	377	96.39582
49 _[aa1]	435	373	96.98932
50	285	374	95.86452

Average No. of Iterations Taken by Our Heuristic	:	316
Average No. of Iterations Taken by CPLEX	:	377
Average Percentage Closeness to the Optimal Solution	:	96.6 %

TABLE 4.7**PROBLEM SIZE : 150 x 150 x 150**

S. No.	No. of Iteration Taken by our Heuristic to give the Solution	No. of Iterations Taken by CPLEX optimization Package to give the Optimal Solution	Percentage Closeness to the Optimal Solution by our Heuristic
1	339	601	98.14
2	328	419	93.05
3	396	635	98.79
4	356	598	98.61
5	352	588	98.71
6	388	737	99.29
7	325	595	98.80
8	337	628	99.14
9	364	895	100.27
10	367	591	98.44
11	354	806	98.89
12	393	472	98.87
13	373	472	98.87
14	395	659	97.07
15	380	697	97.24
16	370	509	98.84
17	353	547	97.08
18	359	472	96.72
19	377	584	97.73
20	380	549	95.90
21	371	734	96.68
22	388	513	97.89
23	354	731	97.99
24	393	622	97.58
25	384	685	97.43

TABLE 4.7 [Contd]

S. No.	No. of Iteration Taken by our Heuristic to give the Solution	No. of Iterations Taken by CPLEX optimization Package to give the Optimal Solution	Percentage Closeness to the Optimal Solution by our Heuristic
26	351	697	96.72
27	377	734	97.78
28	365	584	96.37
29	366	622	97.43
30	381	584	96.37
31	395	510	97.07
32	358	735	97.78
33	370	697	96.72
34	388	659	98.51
35	377	734	98.47
36	383	584	96.66
37	392	472	97.28
38	373	713	95.73
39	364	553	97.59
40	381	665	97.48
41	370	622	97.07
42	383	659	99.42
43	370	473	96.05
44	393	549	92.17
45	359	697	99.18
46	380	673	96.72
47	377	624	97.43
48	381	697	96.72
49	359	472	98.87
50	383	628	98.81

Average No. of Iterations Taken by our Heuristic : 371
 Average No. of Iterations Taken by CPLEX : 620
 Average Percentage Closeness to the Optimal : 97.5 %

TABLE 4.8**PROBLEM SIZE : 200 x 200 x 200**

S. No.	No. of Iteration Taken by our Heuristic to give the Solution	No. of Iterations Taken by CPLEX optimization Package to give the Optimal Solution	Percentage Closeness to the Optimal Solution by our Heuristic
1	568	1458	99.79
2	513	1702	100.60
3	435	1601	99.56
4	690	1415	99.40
5	555	1741	100.00
6	727	1645	99.88
7	625	1299	100.26
8	570	1388	99.59
9	787	1846	100.12
10	767	1371	99.96
11	625	1705	99.43
12	826	1366	99.09
13	762	1820	98.63
14	491	1625	99.46
15	781	1690	100.15
16	671	1495	100.21
17	789	1755	99.65
18	644	1560	100.13
19	837	1820	100.74
20	545	1690	101.50
21	745	1625	100.36
22	880	1560	99.24
23	490	1495	102.03
24	690	1493	99.33
25	608	1560	99.67

TABLE 4.8 [Contd]

S. No.	No. of Iteration Taken by our Heuristic to give the Solution	No. of Iterations Taken by CPLEX optimization Package to give the Optimal Solution	Percentage Closeness to the Optimal Solution by our Heuristic
26	673	1820	100.07
27	873	1625	99.47
28	757	1823	98.95
29	674	1365	99.18
30	555	1560	99.24
31	878	1755	99.61
32	491	1629	99.46
33	672	1560	101.04
34	872	1365	99.15
35	708	1820	98.95
36	508	1560	99.24
37	735	1690	101.50
38	645	1893	100.07
39	809	1690	98.81
40	709	1560	99.24
41	800	1625	100.36
42	799	1690	100.60
43	491	1365	99.16
44	717	1625	100.36
45	682	1430	99.98
46	532	1820	100.74
47	626	1627	100.36
48	757	1690	101.51
49	631	1495	99.31
50	777	1389	99.83

Average No. of Iterations Taken by our Heuristic	:	680
Average No. of Iterations Taken by CPLEX	:	1604
Average Percentage Closeness to the Optimal Result	:	99.3 %

4.5 A Note on the No. Of Iterations Taken by Our Heuristic :

As seen from the Tables given in the previous pages as well as the charts attached at suitable places we can very well see that the number of iterations taken by our solution procedure is considerably low than that was obtained from the CPLEX even for the large size problems .

Though for the small and medium size problem the number of iterations coming from our solution procedure is slightly lower than that of the CPLEX , the clear picture in the difference in number of iterations taken can be seen in large size problems

- For problem size $100 \times 100 \times 100$, it takes iterations 0.75-0.80 (on average) times the iterations taken by the CPLEX ;
- For problem size $150 \times 150 \times 150$, it takes iterations 0.50-0.60 (on average) times the iterations taken by the CPLEX ;
- For problem size $200 \times 200 \times 200$, it takes iterations 0.40-0.50 (on average) times the iterations taken by the CPLEX ;

Thus we can see that our solution heuristic procedure come to be efficient in its solution as well as in the number of iterations taken by it when compared with the CPLEX optimizing package .

4.6 Running our Solution Procedure in $O(c n^2)$ number of steps :

Though our solution procedure runs in $O(c n^3)$ number of steps, now we show the results coming from our heuristic procedure if we run it in $O(c n^2)$ number of steps. As already explained in chapter three all but Heuristic H3 [3.5.6] runs in either $O(n)$ times or in $O(n^2)$ times. Only Heuristic H3 [3.5.6] runs in $O(n^3)$ times. So if we eliminate the improvement in our solution coming from heuristic H3, i.e. if we don't include improvements by H3, we can successfully run our heuristic procedure in $O(c n^2)$ number of times.

We solved 150 problems of sizes varying from $100 \times 100 \times 100$ to $200 \times 200 \times 200$ to show the result coming from running our solution procedure both in $O(c n^2)$ and $O(c n^3)$ times. In the Tables 4.9 - 4.11, percentage closeness to the optimal solution both by running the solution procedure in $O(c n^2)$ and $O(c n^3)$ times are shown.

The result shows that even in running the heuristic in $O(c n^2)$ times, we get good results which are also very close to the optimal solution.

TABLE 4.9
Problem Size : 100 x 100 x 100

S.N.	Solution After H2 only Running in $O(cn^2)$		Solution After H2+H3 Running in $O(cn^3)$		Iterations taken by CPLEX
	Iterations	Percentage Closeness	Iterations	Percentage Closeness	
1	248	97.22	307	97.31	535
2	335	96.43	353	96.47	534
3	311	96.64	337	96.68	380
4	263	95.98	303	96.08	400
5	298	96.92	329	96.96	377
6	284	97.05	289	97.07	390
7	224	96.99	264	97.05	431
8	248	96.03	268	96.06	419
9	298	97.09	315	97.11	586
10	347	96.67	373	96.71	401
11	272	96.20	307	96.26	423
12	283	96.43	286	96.43	365
13	218	96.61	269	96.66	367
14	227	95.98	291	96.10	422
15	221	96.77	277	96.88	468
16	251	96.79	302	96.90	613
17	326	96.91	341	96.94	531
18	223	96.66	274	96.72	380
19	224	96.45	323	96.66	435
20	371	96.36	425	96.44	406
21	400	96.44	463	96.52	521
22	187	96.68	227	96.76	601
23	197	96.69	216	96.72	429
24	187	96.89	204	96.91	484
25	268	96.81	297	96.86	399

CENTRAL LIBRARY
I. I. T. KANPUR
127913

TABLE 4.9 [Contd....]

S.N.	Solution After H2 only Running in $O(cn^2)$		Solution After H2+H3 Running in $O(cn^3)$		Iterations taken by CPLEX
	Iterations	Percentage Closeness	Iterations	Percentage Closeness	
26	275	96.44	342	96.52	384
27	305	96.15	333	96.20	407
28	318	96.63	331	96.67	454
29	287	96.11	298	96.14	376
30	242	96.31	277	96.35	413
31	267	97.06	288	97.09	589
32	233	96.43	253	96.46	378
33	276	96.68	300	96.73	374
34	262	96.89	307	96.96	392
35	251	96.56	293	96.67	400
36	315	95.29	331	95.30	373
37	263	97.05	285	97.09	456
38	238	96.80	279	96.86	574
39	241	96.80	265	96.83	418
40	246	96.48	283	96.55	365
41	288	96.78	310	96.81	400
42	310	97.26	354	97.32	497
43	236	97.07	317	97.17	617
44	261	96.85	319	97.00	398
45	357	96.55	383	96.59	503
46	291	96.41	342	96.51	397
47	186	96.94	263	97.06	380
48	237	96.65	278	96.72	378
49	258	96.60	304	96.69	559
50	274	96.53	293	96.57	363

TABLE 4.10
Problem Size : 150 x 150 x 150

S.N.	Solution After H2 only Running in $O(cn^2)$		Solution After H2+H3 Running in $O(cn^3)$		Iterations taken by CPLEX
	Iterations	Percentage Closeness	Iterations	Percentage Closeness	
1	357	98.43	360	98.45	651
2	360	98.39	371	98.51	596
3	336	99.18	341	99.23	578
4	367	98.10	378	98.35	655
5	339	98.35	345	98.45	691
6	371	97.92	383	98.45	562
7	348	99.03	353	99.08	603
8	405	99.14	410	99.21	616
9	323	99.31	342	99.51	583
10	372	98.20	383	99.21	623
11	352	99.45	369	99.73	604
12	400	99.67	406	99.73	626
13	386	99.46	388	99.51	565
14	366	99.55	376	99.67	582
15	343	98.63	357	98.82	613
16	362	98.56	392	98.96	565
17	313	99.55	320	99.61	640
18	328	95.39	351	96.04	716
19	380	98.52	380	98.52	578
20	384	99.39	392	99.50	618
21	351	99.06	351	99.06	561
22	375	99.09	384	99.16	641
23	343	99.21	343	99.21	575
24	384	99.29	392	99.40	636
25	328	98.64	340	98.75	675

TABLE 4.10 [Contd...]

S.N.	Solution After H2 only Running in $O(cn^2)$		Solution After H2+H3 Running in $O(cn^3)$		Iterations taken by CPLEX
	Iterations	Percentage Closeness	Iterations	Percentage Closeness	
26	309	97.88	325	98.03	595
27	355	98.89	379	99.34	535
28	381	99.58	381	99.58	573
29	346	99.46	362	99.69	587
30	347	99.09	364	99.33	595
31	372	98.86	377	98.96	635
32	370	99.21	378	99.30	658
33	380	98.32	401	98.59	604
34	383	98.80	383	98.80	707
35	362	99.71	362	99.71	624
36	345	98.98	369	99.38	582
37	379	98.95	399	99.12	644
38	376	99.77	388	99.92	601
39	375	99.09	382	99.18	577
40	362	99.03	366	99.17	610
41	364	98.95	370	99.04	613
42	326	98.76	354	99.28	619
43	356	98.94	356	98.94	606
44	346	99.46	362	99.69	587
45	370	99.09	375	99.20	567
46	369	99.18	384	99.41	593
47	318	96.79	326	97.59	595
48	367	98.50	383	98.72	589
49	384	95.63	386	97.36	596
50	388	99.57	393	99.63	597

TABLE 4.11
Problem Size : 200 x 200 x 200

S.N.	Solution After H2 only Running in $O(cn^2)$		Solution After H2+H3 Running in $O(cn^3)$		Iterations taken by CPLEX
	Iterations	Percentage Closeness	Iterations	Percentage Closeness	
1	289	99.98	289	99.98	1281
2	711	99.19	711	99.19	1599
3	283	99.97	283	99.97	1620
4	521	99.96	521	99.96	1850
5	623	99.86	623	99.86	1762
6	457	99.76	457	99.76	1569
7	711	99.88	711	99.88	1497
8	758	99.88	758	99.88	1260
9	480	99.19	480	99.19	1315
10	410	99.88	410	99.88	1554
11	815	99.68	815	99.68	1596
12	270	99.96	270	99.96	1463
13	494	99.16	494	99.16	1159
14	446	99.96	446	99.96	1315
15	503	99.53	503	99.53	1346
16	917	99.67	917	99.67	1441
17	694	99.65	694	99.65	1340
18	275	99.98	275	99.98	1286
19	930	99.29	930	99.29	1780
20	452	99.65	452	99.65	1410
21	569	99.85	569	99.85	1603
22	535	99.66	535	99.66	1328
23	364	99.98	364	99.98	1320
24	312	99.78	312	99.78	1618
25	375	99.52	375	99.52	1330

TABLE 4.11 [Contd...]

S.N.	Solution After H2 only Running in $O(cn^2)$		Solution After H2+H3 Running in $O(cn^3)$		Iterations taken by CPLEX
	Iterations	Percentage Closeness	Iterations	Percentage Closeness	
26	787	99.98	787	99.98	1266
27	478	99.56	478	99.56	1432
28	476	99.66	476	99.66	1413
29	502	99.67	502	99.67	1307
30	417	99.98	417	99.98	1803
31	345	99.99	345	99.99	1830
32	299	99.48	299	99.48	1817
33	361	99.54	361	99.54	1619
34	539	99.64	539	99.64	1462
35	628	99.96	628	99.96	1424
36	890	99.55	890	99.55	1266
37	201	99.65	201	99.65	1281
38	284	99.81	284	99.81	1442
39	439	99.39	439	99.39	1263
40	201	99.65	201	99.65	1466
41	495	99.66	495	99.66	1370
42	616	99.67	616	99.67	1245
43	845	99.66	845	99.66	1388
44	628	99.64	628	99.64	1354
45	628	99.66	628	99.66	1366
46	283	99.61	283	99.61	1361
47	585	99.76	585	99.76	1344
48	591	99.55	591	99.55	1241
49	486	99.57	486	99.57	1571
50	583	99.51	583	99.51	1381

CHAPTER 5

CONCLUSIONS AND DIRECTIONS FOR FUTURE RESEARCH

As far as the computational complexity of the solution procedure given by us is concerned , we can say that our heuristic runs in $O(c n^3)$ number of step . Using this solution procedure we solved transshipment problems of small as well as large size . It is found that even for the large sized problems the solution comes closer to 97 % to the optimal solution obtained by CPLEX . We also showed that even while running the heuristic in $O(c n^2)$ times , the solution procedure gives good results . Thus the solution coming from our solution procedure can give a good start to conventional dual based algorithms to reach the optimal solution.

We hope that future work would extend our work to multistage minimum cost flow problems with no restrictions on capacity of flow on arcs. May be later work can concentrate on capacitated network flows problem as well

REFERENCES

1. Aashtiani, H.A , and Magnanti, T.L., "Implementing Primal Dual Network Flow Algorithms" , *Technical Report OR*, 1976, 55-76, Operation Research Center, M.I.T. , Cambridge, M A.
2. Ahuja, R.K., Magnanti, T.L., and Orlin, J.B. 1993. *Network flows : Theory , Algorithms and Applications*, Prentice-Hall, Englewood Cliffs, New Jersey (US).
3. Ahuja, R K., Goldberg, A V., Orlin, J.B., Tarjan, R.E., "Finding Minimum Cost Flows by Double Scaling" , *Mathematical Programming*, 1992, 53, 243-266 .
4. Ahuja, R K. and Orlin, J.B., "Distance Directed Augmenting Path Algorithms for the Maximum Flow and Parametric Maximum Flow Problems" , *Naval Research Logistics Quaterly*, 1991,38, 413-430 .
5. Edmonds, J., and Karp, R.M., "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problem" , *Journal of ACM*, 1972,19, 248-264.
6. Fulkerson, D.R., "An Out-of-Kilter Method for Minimum Cost Flow Problem" , *SIAM Journal of Applied Mathematics* 9,1961, 18-27.
7. Gabow, H.N., "Scaling Algorithms for Metwork Problems" , *Journal of Computer and Systems Sciences*, 1985, 31, 148-168.
8. Galil, Z , and Tardos,E , "An $O((m+n \log n) n^2 \log n)$ Minimum Cost Flow Algorithms", *proceedings 27th Annual IEEE Symposium on Foundations of Computer Sciences*, 1986, 1-9.
9. Goldberg, A.V., and Tarjan, R.E., "Solving Minimum Cost Flow Problem by Successive Approximations", *Proceedings 19th ACM Symposium on Theory of Computing*, May, 1987.

10. Grigoriadis, M D., "An Efficient Implementation of the Network Simplex Method" , *Mathematical Programming*, 1986,26, 83-111.
11. Hassin, R , "The Minimum Cost Flow Problem : A unifying Approach to Dual Algorithm and A New Tree Search Algorithm" , *Mathematical Programming*, 1983, 25, 228-239
12. Hassin, R., "Algorithm for the Minimum Cost Circulation Problem Based on Maximizing the Mean Improvement", *Operation Research Letter*, 1992, 12, 227-233.
13. Helagson, R.V., and Kenington, J.L., "An Efficient Procedure for Implementing a Dual Simplex Network Flow Algorithm" *AIEE Trans.*, 1977, 9 , 63-68.
14. Karmakar, N., "A New Polynomial-Time Algorithm for Linear Programming", *Combinatorica* , 1984, 4, 373-395.
15. Klingman, D., Napier, K., and Stitz, J., "A Program for Generating Large Scale Capacitated Assignment, Transportation, and Minimum Cost Flow Problems", *Management Science*, 1974, 20, 814-821.
16. Orlin, J.B., Plotkin, A S., and tardos, E., :Polynomial Dual Network Simplex Algorithms" , *Mathematical Programming*, 1993, 60, 255-276.
17. Orlin, J.B., "Genuinely Polynomial Simplex and Non-singular Algorithms for the Minimum Cost Flow Problem" , *Technical Report No. 1615-84*, Sloan School of Management, MIT (Cambridge, MA, 1984)
18. Orlin, J.B., "A Faster Strongly Polynomial Cost Flow Algorithm", *Operations Research*, 1993, vol. 41, 338-350.
19. Orden, A., "The Transshipment Problem", *Management Science*, 1956, 2, 276-285.

20. Armstrong, R.D., Klingman, D., Whitman, D., "Implementation and Analysis of a Variant of the Dual Method for the Capacitated Transshipment Problem", *European Journal of Operational Research*, 1980, 4, 403-420.
21. Glover, F., Karney, D., Klingman, D., and Russel, R., "Solving Singly Contrained Transshipment Problems", *Transportation Science*, 1978, vol. 12 No. 4, 277-297.
22. Glover, F., Karney, D., Klingman, D., Napier, A., "A computational Study on Start Procedures, Basis Change Criteria, and Solution Algorithms for Transportation Problems", *Management Science*, 1974, 20, 793-813 .
23. Masch, V V , "The Cyclic Method of Solving the Transshipment Problem with an Additional Linear Constraint", *Networks*, 1980, 10, 17-31.
24. Winston, W.L 1994 *Operations Research - Application and Algorithms*, Duxbury Press, An Imprint of Wad Sworth Publishing Company.
25. Sharma, R.R.K., "Food grains distribution in the Indian context : An Operational Study" , Project Sponsored by the Department of Science and Technology, Govt. of India, New Delhi. D.O. No SR/0Y/E-01/90, Aug. 1993.
26. Sharma, K.D., "A Dual Based Approach for the Simple Transportation Problem", unpublished M.Tech thesis, Department of Industrial and Management Engineering, I.I T., Kanpur, India, 1998.

Some Solved Numerical Problems Using the Heuristic

In order to fully illustrate the working of the solution procedure given by us in this paper , we are here giving three sample problems .

Problem 1 :

Problem Size : 3 X 3 X 3

Given :

Supplies $b_i = (0.2, 0.5, 0.3)$

Demands $d_k = (0.25, 0.4, 0.35)$

$$\text{Matrix } C1_{ij} = \begin{bmatrix} 3 & \dots & 5 & \dots & 8 \\ 9 & \dots & 6 & \dots & 1 \\ 2 & \dots & 3 & \dots & 8 \end{bmatrix}$$

$$\text{Matrix } C2_{jk} = \begin{bmatrix} 2 & \dots & 1 & \dots & 4 \\ 3 & \dots & 2 & \dots & 5 \\ 6 & \dots & 7 & \dots & 6 \end{bmatrix}$$

Solution Procedure :

Iteration # 0 :

$$C1_{1j} = (\underline{3}, 5, 8)$$

$$C2_{j1} = (\underline{2}, 3, 6)$$

$$C1_{2j} = (9, 6, \underline{1})$$

$$C2_{j2} = (\underline{1}, 2, 7)$$

$$C1_{3j} = (\underline{2}, 3, 8)$$

$$C2_{j3} = (\underline{4}, 5, 6)$$

$$C1_i^* = (3, 1, 2)$$

$$C2_k^* = (2, 1, 4)$$

$$m_1 = \{1\}$$

$$l_1 = \{1\}$$

$$m_2 = \{3\}$$

$$l_2 = \{1\}$$

$$m_3 = \{1\}$$

$$l_3 = \{1\}$$

Thus we have ,

$$V1_0 = 3$$

$$V2_0 = 4$$

$$z_1 = 0, z_2 = 2, z_3 = 1$$

$$v_1 = 2, v_2 = 3, v_3 = 0$$

$$\begin{aligned} Z_{\text{dual}} &= (V1_0 - \sum_{i=1}^I z_i \cdot b_i) + (V2_0 - \sum_{k=1}^K v_k \cdot d_k) \\ &= 4.00 \end{aligned}$$

Iteration # 1

$$WJ = l_1 = \{ 1 \} \quad \text{using H2}$$

$$\text{gain} = d_1 + d_2 + d_3 = 1.0 \quad \text{loss} = b_1 + b_3 = 0.5$$

$$\text{Net_Benefit} = \text{gain} - \text{loss} = 0.5 \quad [> 0 \text{ Therefore Increment Possible }]$$

$$\text{Extent_of_Increase2} = \min (1, 1, 1) = 1$$

$$\text{Extent_of_decrease1} = \min (\infty, 8, \infty) = 8$$

$$\text{Extntet_of_Increase} = \min (\text{Extent_of_Increase2}, \text{Extent_of_Decrease2}) = 1$$

$$C1_{1j} = (2, 5, 8) \quad C2_{j1} = (\underline{3}, \underline{3}, 6)$$

$$C1_{2j} = (8, 6, \underline{1}) \quad C2_{j2} = (\underline{2}, \underline{2}, 7)$$

$$C1_{3j} = (\underline{1}, 3, 8) \quad C2_{j3} = (\underline{5}, \underline{5}, 6)$$

$$C1_i^* = (2, 1, 1) \quad C2_k^* = (3, 2, 5)$$

$$m_1 = \{1\} \quad l_1 = \{1, 2\}$$

$$m_2 = \{3\} \quad l_2 = \{1, 2\}$$

$$m_3 = \{1\} \quad l_3 = \{1, 2\}$$

Thus we have ,

$$V1_0 = 2 \quad V2_0 = 5$$

$$z_1 = 0, z_2 = 1, z_3 = 1 \quad v_1 = 2, v_2 = 3, v_3 = 0$$

$$\begin{aligned} Z_{\text{dual}} &= (V1_0 - \sum_{i=1}^I z_i \cdot b_i) + (V2_0 - \sum_{k=1}^K v_k \cdot d_k) \\ &= 4.50 \end{aligned}$$

Iteration # 2

$$WJ = l_1 = \{ 1, 2 \} \quad \text{using H2}$$

$$\text{gain} = d_1 + d_2 + d_3 = 1.0 \quad \text{loss} = b_1 + b_3 = 0.5$$

$$\text{Net_Benefit} = \text{gain} - \text{loss} = 0.5 \quad [> 0 \text{ Therefore Increment Possible }]$$

$$\text{Extent_of_Increase2} = \min(3, 5, 1) = 1$$

$$\text{Extent_of_decrease1} = \min(-\infty, 5, \infty) = 5$$

$$\text{Extnet_of_Increase} = \min(\text{Extent_of_Increase2}, \text{Extent_of_Decrease2}) = 1$$

$$C1_{1j} = (1, 4, 8) \quad C2_{j1} = (4, 4, 6)$$

$$C1_{2j} = (7, 5, 1) \quad C2_{j2} = (3, 3, 7)$$

$$C1_{3j} = (0, 2, 8) \quad C2_{j3} = (6, 6, 6)$$

$$C1_i^* = (1, 1, 0) \quad C2_k^* = (4, 3, 6)$$

$$m_1 = \{1\} \quad l_1 = \{1, 2\}$$

$$m_2 = \{3\} \quad l_2 = \{1, 2\}$$

$$m_3 = \{1\} \quad l_3 = \{1, 2, 3\}$$

Thus we have ,

$$V1_0 = 1 \quad V2_0 = 6$$

$$z_1 = 0, z_2 = 0, z_3 = 1 \quad v_1 = 2, v_2 = 3, v_3 = 0$$

$$\begin{aligned} Z_{\text{dual}} &= (V1_0 - \sum_{i=1}^J z_i \cdot b_i) + (V2_0 - \sum_{k=1}^K v_k \cdot d_k) \\ &= 5.00 \end{aligned}$$

Iteration # 3 :

$$WJ = l_1 = \{1, 2\} \quad \text{using H2}$$

$$\text{gain} = d_1 + d_2 = 0.65 \quad \text{loss} = b_1 + b_3 = 0.5$$

$$\text{Net_Benefit} = \text{gain} - \text{loss} = 0.15 \quad [>0 \text{ Therefore Increment Possible}]$$

$$\text{Extent_of_Increase2} = \min(2, 3, \infty) = 2$$

$$\text{Extent_of_decrease1} = \min(-\infty, 4, \infty) = 4$$

$$\text{Extnet_of_Increase} = \min(\text{Extent_of_Increase2}, \text{Extent_of_Decrease2}) = 2$$

$$C1_{1j} = (-1, 2, 8) \quad C2_{j1} = (6, 6, 6)$$

$$C1_{2j} = (5, 3, 1) \quad C2_{j2} = (5, 5, 7)$$

$$C1_{3j} = (-2, 0, 8) \quad C2_{j3} = (8, 8, 6)$$

$$C1_i^* = (-1, 1, -2) \quad C2_k^* = (6, 5, 6)$$

$$m_1 = \{1\} \quad l_1 = \{1, 2, 3\}$$

$$m_2 = \{3\}$$

$$l_2 = \{1,2\}$$

$$m_3 = \{1\}$$

$$l_3 = \{3\}$$

Thus we have ,

$$V1_0 = 1$$

$$V2_0 = 6$$

$$z_1 = 2, z_2 = 0, z_3 = 3$$

$$v_1 = 0, v_2 = 1, v_3 = 0$$

$$Z_{\text{dual}} = (V1_0 - \sum_{i=1}^I z_i \cdot b_i) + (V2_0 - \sum_{k=1}^K v_k \cdot d_k)$$

$$= 5.30$$

Iteration # 4

$$WJ = l_1 = \{1,2\} \quad \text{using H2}$$

$$\text{gain} = d_1 + d_2 + d_3 = 1.00$$

$$\text{loss} = b_1 + b_2 + b_3 = 1.00$$

$$\text{Net_Benefit} = \text{gain} - \text{loss} = 0 \quad [\text{Therefore No Increment Possible}]$$

Other combinations of the current l_k 's also using H2 shows $\text{Net_Benefit} \leq 0$, hence now we go for finding WJ using H3.

Iteration # 5 :

$$WJ = \{1,2,3\} \quad \text{using H3}$$

$$\text{gain} = \text{loss} = 1.0 \quad \text{Hence no improvement possible}$$

Other combinations of l_k 's also show $\text{Net_benefit} \leq 0$, hence no improvement possible at all.

Heuristic Terminates Here

$$\text{Objective function Value } Z_{\text{dual}} = 5.30$$

$$\text{No. of iterations taken} = 5$$

$$\text{Optimal Results From CPLEX} = 5.30$$

Problem 2 :

Problem Size : 4 X 4 X 4

Given

Supplies $b_i = (0.1, 0.3, 0.5, 0.1)$

Demands $d_k = (0.2, 0.3, 0.4, 0.1)$

$$\text{Matrix } C1_{ij} = \begin{bmatrix} 3 & 2 & 5 & 6 \\ 4 & 3 & 4 & 2 \\ 2 & 5 & 8 & 6 \\ 7 & 6 & 2 & 1 \end{bmatrix}$$

$$\text{Matrix } C2_{jk} = \begin{bmatrix} 2 & 3 & 7 & 8 \\ 6 & 9 & 5 & 3 \\ 1 & 3 & 2 & 4 \\ 4 & 3 & 2 & 3 \end{bmatrix}$$

Solution Procedure :

Iteration # 0 :

$$\begin{aligned} C1_{1j} &= (3, \underline{2}, 5, 6) & C2_{j1} &= (2, 6, \underline{1}, 4) \\ C1_{2j} &= (4, 3, 4, \underline{2}) & C2_{j2} &= (\underline{3}, 9, \underline{3}, \underline{3}) \\ C1_{3j} &= (\underline{2}, 5, 8, 6) & C2_{j3} &= (7, 5, \underline{2}, \underline{2}) \\ C1_{4j} &= (7, 6, 2, \underline{1}) & C2_{j4} &= (8, \underline{3}, 4, \underline{3}) \\ C1_i^* &= (2, 2, 2, 1) & C2_k^* &= (1, 3, 2, 3) \\ m_1 &= \{2\} & l_1 &= \{3\} \\ m_2 &= \{4\} & l_2 &= \{1, 3, 4\} \\ m_3 &= \{1\} & l_3 &= \{3, 4\} \\ m_4 &= \{4\} & l_4 &= \{2, 4\} \end{aligned}$$

Thus we have,

$$\begin{aligned} V1_0 &= 2 & V2_0 &= 3 \\ z_1 = 0, z_2 = 0, z_3 = 0, z_4 = 1 & & v_1 = 2, v_2 = 0, v_3 = 1, v_4 = 0 \end{aligned}$$

$$Z_{\text{dual}} = (V1_0 - \sum_{i=1}^I z_i \cdot b_i) + (V2_0 - \sum_{k=1}^K v_k \cdot d_k)$$

$$= 4.10$$

Iteration # 1

$$WJ = l_1 = \{ 3 \} \quad \text{using H2}$$

$$\text{gain} = d_1 = 0.2 \quad \text{loss} = 0$$

$$\text{Net_Benefit} = \text{gain} - \text{loss} = 0.2 \quad [> 0 \text{ Therefore Increment Possible }]$$

$$\text{Extent_of_Increase2} = \min (1, \infty, \infty, \infty) = 1$$

$$\text{Extent_of_decrease1} = \min (3, 2, 6, 1) = 1$$

$$\text{Extnet_of_Increase} = \min (\text{Extent_of_Increase2}, \text{Extent_of_Decrease2}) = 1$$

$$C1_{1j} = (3, \underline{2}, 4, 6) \quad C2_{j1} = (\underline{2}, 6, \underline{2}, 4)$$

$$C1_{2j} = (4, 3, 3, \underline{2}) \quad C2_{j2} = (\underline{3}, 9, 4, \underline{3})$$

$$C1_{3j} = (\underline{2}, 5, 7, 6) \quad C2_{j3} = (7, 5, 3, \underline{2})$$

$$C1_{4j} = (7, 6, \underline{1}, \underline{1}) \quad C2_{j4} = (8, \underline{3}, 5, \underline{3})$$

$$C1_i^* = (2, 2, 2, 1) \quad C2_k^* = (2, 3, 2, 3)$$

$$m_1 = \{2\} \quad l_1 = \{1, 3\}$$

$$m_2 = \{4\} \quad l_2 = \{1, 4\}$$

$$m_3 = \{1\} \quad l_3 = \{4\}$$

$$m_4 = \{3, 4\} \quad l_4 = \{2, 4\}$$

Thus we have,

$$V1_0 = 2 \quad V2_0 = 3$$

$$z_1 = 0, z_2 = 0, z_3 = 0, z_4 = 1 \quad v_1 = 1, v_2 = 0, v_3 = 1, v_4 = 0$$

$$Z_{\text{dual}} = (V1_0 - \sum_{i=1}^I z_i \cdot b_i) + (V2_0 - \sum_{k=1}^K v_k \cdot d_k)$$

$$= 4.30$$

Iteration # 2 :

$$WJ = l_1 = \{ 1, 3 \} \quad \text{using H2}$$

$$\text{gain} = d_1 = 0.2 \quad \text{loss} = b_3 + b_4 = 0.6$$

$$\text{Net_Benefit} = \text{gain} - \text{loss} = -0.4 \quad [< 0 \text{ Therefore No Increment Possible }]$$

Other combinations of the current l_k 's also show $\text{Net_Benefit} \leq 0$ using H2

and hence we now go to find WJ using H3.

Iteration # 3 :

$WJ = \{ 1, 3, 4 \}$ using H3.

$$\text{gain} = d_1 + d_2 + d_3 = 0.9 \quad \text{loss} = b_2 + b_3 + b_4 = 0.9$$

$\text{Net_Benefit} = 0$ Therefore no Improvement possible.

Other combinations of the current l_k 's also show no improvement using H3.

Hence no improvement at all

Heuristic terminates here

Objective function value $Z_{\text{dual}} = 4.30$

No of Iterarions taken = 3

Optimal Result from CPLEX = 4.30

Problem 3 :

Problem Size : 3 X 4 X 3

Given .

Supplies $b_i = (0.3, 0.5, 0.2)$

Demands $d_k = (0.1, 0.7, 0.2)$

$$\text{Matrix } C1_{ij} = \begin{bmatrix} 3 & 4 & 2 & 3 \\ 5 & 6 & 4 & 1 \\ 3 & 2 & 5 & 3 \end{bmatrix}$$

$$\text{Matrix } C2_{jk} = \begin{bmatrix} 3 & 2 & 5 \\ 6 & 1 & 3 \\ 2 & 1 & 4 \\ 4 & 3 & 1 \end{bmatrix}$$

Solution Procedure :

Iteration # 0 :

$$C1_{1j} = (3, 4, \underline{2}, 3)$$

$$C2_{j1} = (3, 6, \underline{2}, 4)$$

$$C1_{2j} = (5, 6, 4, \underline{1})$$

$$C2_{j2} = (2, \underline{1}, \underline{1}, 3)$$

$$C1_{3j} = (3, \underline{2}, 5, 3)$$

$$C2_{j3} = (5, 3, 4, \underline{1})$$

$$C1_i^* = (2, 1, 2)$$

$$C2_k^* = (2, 1, 1)$$

$$m_1 = \{3\}$$

$$l_1 = \{3\}$$

$$m_2 = \{4\}$$

$$l_2 = \{2, 3\}$$

$$m_3 = \{2\}$$

$$l_3 = \{4\}$$

Thus we have,

$$V1_0 = 2$$

$$V2_0 = 2$$

$$z_1 = 0, z_2 = 1, z_3 = 0$$

$$v_1 = 0, v_2 = 1, v_3 = 1$$

$$Z_{\text{dual}} = (V1_0 - \sum_{i=1}^I z_i \cdot b_i) + (V2_0 - \sum_{k=1}^K v_k \cdot d_k)$$

$$= 2.60$$

Iteration #1

$$WJ = l_1 = \{3\} \quad \text{using H2}$$

$$\text{gain} = d_1 = 0.1$$

$$\text{loss} = b_3 = 0.3$$

$$\text{Net_Benefit} = \text{gain} - \text{loss} = -0.2 \quad [< 0 \text{ Therefore No Increment Possible }]$$

Iteration # 2 :

$$WJ = l_2 = \{2, 3\} \quad \text{using H2.}$$

$$\text{gain} = d_1 + d_2 = 0.8$$

$$\text{loss} = b_1 + b_3 = 0.5$$

$$\text{Net_Benefit} = \text{gain} - \text{loss} = 0.3 \quad [> 0 \text{ Hence Improvement possible}]$$

$$\text{Extent_of_Increase2} = \min(\infty, 1, \infty) = 1$$

$$\text{Extent_of_decrease1} = \min(\infty, 3, \infty) = 3$$

$$\text{Extnet_of_Increase} = \min(\text{Extent_of_Increase2}, \text{Extent_of_Decrease2}) = 1$$

$$C1_{1j} = (3, 3, \underline{1}, 3)$$

$$C2_{j1} = (\underline{3}, 7, \underline{3}, 4)$$

$$C1_{2j} = (5, 5, 3, \underline{1})$$

$$C2_{j2} = (\underline{2}, \underline{2}, \underline{2}, 3)$$

$$C1_{3j} = (3, \underline{1}, 4, 3)$$

$$C2_{j3} = (5, 4, 5, \underline{1})$$

$$C1_i^* = (1, 1, 1)$$

$$C2_k^* = (3, 2, 1)$$

$$m_1 = \{3\}$$

$$l_1 = \{1, 3\}$$

$$m_2 = \{4\}$$

$$l_2 = \{1, 2, 3\}$$

$$m_3 = \{2\}$$

$$l_3 = \{4\}$$

Thus we have,

$$V1_0 = 1$$

$$V2_0 = 3$$

$$z_1 = 0, z_2 = 0, z_3 = 0$$

$$v_1 = 0, v_2 = 1, v_3 = 2$$

$$Z_{\text{dual}} = (V1_0 - \sum_{i=1}^I z_i \cdot b_i) + (V2_0 - \sum_{k=1}^K v_k \cdot d_k)$$

$$= 2.90$$

Iteration # 3 .

$$WJ = l_1 = \{ 1, 3 \} \quad \text{using H2}$$

$$\text{gain} = d_1 = 0.1 \quad \text{loss} = b_1 = 0.3$$

$$\text{Net_Benefit} = \text{gain} - \text{loss} = -0.2 \quad [< 0 \text{ Therefore No Increment Possible }]$$

Iteration # 4

$$WJ = l_2 = \{ 1, 2, 3 \}$$

$$\text{gain} = d_1 + d_2 = 0.8 \quad \text{loss} = b_1 + b_3 = 0.5$$

$$\text{Net_Benefit} = \text{gain} - \text{loss} = 0.3 \quad [> 0 \text{ Therefore Improvement possible}]$$

$$\text{Extent_of_Increase2} = \min (\infty, 1, \infty) = 1$$

$$\text{Extent_of_Decrease1} = \min (\infty, 2, \infty) = 2$$

$$\text{Extent_of_Increase} = \min (\text{Extent_of_Increase2}, \text{Extent_of_Decrease2}) =$$

1

$$C1_{1j} = (2, 2, \underline{0}, 3) \quad C2_{j1} = (\underline{4}, 8, \underline{4}, 4)$$

$$C1_{2j} = (4, 4, 2, \underline{1}) \quad C2_{j2} = (\underline{3}, \underline{3}, \underline{3}, \underline{3})$$

$$C1_{3j} = (2, \underline{0}, 3, 3) \quad C2_{j3} = (6, 5, 6, \underline{1})$$

$$C1_i^* = (0, 1, 0) \quad C2_k^* = (4, 3, 1)$$

$$m_1 = \{3\} \quad l_1 = \{1, 3, 4\}$$

$$m_2 = \{4\} \quad l_2 = \{1, 2, 3, 4\}$$

$$m_3 = \{2\} \quad l_3 = \{4\}$$

Thus we have,

$$V1_0 = 1 \quad V2_0 = 4$$

$$z_1 = 1, z_2 = 0, z_3 = 1 \quad v_1 = 0, v_1 = 1, v_1 = 3$$

$$Z_{\text{dual}} = (V1_0 - \sum_{i=1}^I z_i \cdot b_i) + (V2_0 - \sum_{k=1}^K v_k \cdot d_k)$$

$$= 3.20$$

Iteration # 5 :

$$WJ = l_1 = \{ 1, 3, 4 \} \quad \text{using H2}$$

$$\text{gain} = d_1 + d_3 = 0.3 \quad \text{loss} = b_1 + b_2 = 0.8$$

$$\text{Net_Benefit} = \text{gain} - \text{loss} = -0.5 \quad [< 0 \text{ Therefore No Increment Possible }]$$

Other combinations of the current l_k 's also using H2 shows $\text{Net_Benefit} \leq 0$, hence now we go for finding WJ using H3.

The combinations of the current l_k 's also show no improvement using H3.
Hence no Improvement at all.

Heuristic terminates here

$$\text{Objective function value } Z_{\text{dual}} = 3.20$$

$$\text{No. of Iterarions taken} = 5$$

$$\text{Optimal Results from CPLEX} = 3.20$$

**Pascal Program made to test the solution procedure given in this paper for
various randomly generated Problems .**

Program Heuristic(Input,Output),

Label 100,200, (** Lables Declaration **)

Const (** Constants Declaration **)

 xx = 200,

 yy = 200,

 zz = 200;

 inf = 100000000,

Type

 Digits = set of 1..200,

Var (** Variables Declaration **)

 over : boolean;

 dec,C1 : array[1..xx,1..yy] of integer;

 C2 : array[1..yy,1..zz] of integer,

 b : array[1..xx] of real,

 d : array[1..zz] of real;

 m : array[1..xx] of digits;

 l : array[1..zz] of digits,

 k1 , w_j : digits;

 u , i , j , k , k2 : integer;

 c11 , c1star : array[1..xx] of integer;

 c22 , c2star : array[1..zz] of integer;

 eoi : array[1..zz] of integer;

 eod : array[1..xx] of integer;

 eoi1 , eod1 , feoi : integer;

```
x , y , z : integer;  
opt , gain , loss , net_benefit  real,  
iterations : integer;
```

Procedure To Initialize all the variables used in the Program

Procedure Initialize,

var

 i,j,k:integer;

Begin

 for i:=1 to xx do

 for j:=1 to yy do

 c1[i,j]:=0;

 for j:=1 to yy do

 for k:=1 to zz do

 c2[j,k]:=0;

 for i:=1 to xx do

 b[i]:=0.0;

 for k:=1 to zz do

 d[k]:=0.0;

 for i:=1 to xx do

 m[i]:=[];

 for k:=1 to zz do

 l[k]:=[],

 gain:=0.0;

 loss:=0.0;

 net_benefit:=0.0;

 iterations:=0;

End;

Procedure to Read Input for the Problem

Procedure Read_input,

var

i, j, k : integer,

Begin

read(x), (** Supply Nodes **)

read(y); (** Intermediate Nodes **)

read(z), (** Demand Nodes **)

for i:=1 to x do

read(b[i]), (** Supplies **)

for k:=1 to z do

read(d[k]), (** Demands **)

for i:=1 to x do

begin

for j:=1 to y do

read(c1[i,j]), (** Unit Transportation Cost **)

readln,

end;

for j:=1 to y do

begin

for k:=1 to z do

read(c2[j,k]), (** Unit Transportation Cost **)

readln,

end;

End;

Procedure to Find the Values of $C1_i^*$ and $C2_k^*$

Procedure Find_Stars,

var

i, j, k : integer,

Begin

for $i:=1$ to x do

begin

$c1star[i]:=c1[i,1];$

for $j:=1$ to y do

begin

if ($c1star[i]>c1[i,j]$) then

$c1star[i]:=c1[i,j];$

end,

end;

for $k=1$ to z do

begin

$c2star[k]:=c2[1,k];$

for $j:=1$ to y do

begin

if ($c2star[k]>c2[j,k]$) then

$c2star[k]:=c2[j,k];$

end,

end;

End;

Procedure to Find the Sets m_i and l_k used in the program

Procedure find_sets_m_and_l,

var

i, j, k . integer,

Begin

 for $i:=1$ to x do

$m[i]:=[]$;

 for $k:=1$ to z do

$l[k]:=[]$;

 for $i:=1$ to x do

 for $j:=1$ to y do

 if ($c1[i,j]=c1star[i]$) then

$m[i]:=m[i]+[j]$;

 for $k:=1$ to z do

 for $j:=1$ to y do

 if ($c2[j,k]=c2star[k]$) then

$l[k]:=l[k]+[j]$;

End;

Procedure to Find Net Benefit

Procedure Find_net_benefit;

var

i, j, k : integer,

Begin

gain:=0.0,

loss:=0.0;

net_benefit:=0.0;

for k:=1 to z do

if (l[k] <= wj) then

gain:=gain+d[k],

for j:=1 to y do

begin

if (j in wj) then

for i:=1 to x do

if (j in m[i]) then

begin

loss:=loss+b[i],

m[i]:=[],

end;

end;

net_benefit:=gain-loss,

End;

Procedure to Find the Extent of Increase possible to Improve the Solution

Procedure Find_extent_of_increase;

var

i, j, k . integer,

Begin

for k:=1 to z do

begin

 eoi[k] :=inf,

 c22[k]:=inf;

end;

for k:=1 to z do

begin

 for j:=1 to y do

 if not(j in wj) then

 if (c22[k]>c2[j,k]) then

 c22[k]:=c2[j,k];

 eoi[k]:=c22[k]-c2star[k],

 if (eoi[k]=0) then

 eoi[k]:=inf;

end;

eoil :=inf,

for k:=1 to z do

 if (eoil>eoi[k]) then

 eoil:=eoi[k];

for i:=1 to x do

 for j:=1 to y do

begin

 dec[i,j]:=inf;

 eod[i]:=inf;

```

        c11[i]:=inf,
    end,
for i:=1 to x do
begin
    for j:=1 to y do
        if (j in wj) then
            begin
                dec[i,j]:=c1[i,j]-c1star[i];
                if (c1[i,j]-c1star[i]=0) then
                    dec[i,j]:=inf,
            end;
        end;
    end;
    i:=1;

while(i<=x) do
begin
    over:=true;
    eod[i]:=dec[i,1];
    for j:=1 to y do
        if (j in wj) then
            begin
                if (dec[i,j]=inf) then
                    begin
                        eod[i]:=inf,
                        over:=false,
                    end
                else
                    if (over) then
                        begin
                            if (eod[i]>dec[i,j]) then
                                eod[i]:=dec[i,j];
                            end;
                        end;
                    end;
            end;
        end;
    end;
end;

```


Procedure to Update the Cost Coefficients after Finding Extent of Increase

Procedure Update_costs;

var i,j,k:integer,

Begin

 find_extent_of_increase;

 for j:=1 to y do

 if (j in wj) then

 begin

 for i:=1 to x do

 c1[i,j]:=c1[i,j]-feoi;

 for k:=1 to x do

 c2[j,k]:=c2[j,k]+feoi;

 end;

 show_best_result,

End;

Main Program Begins Here

Begin

 Initialize;

 Read_Input;

100 :

 for k:=1 to z do

 begin

 u:=u+1;

 find_stars;

 find_sets_m_and_l;

 wj:=l[k];

```

    find_net_benefit;
    if (net_benefit>0) then
    begin
        iterations:=iterations+1;
        update_costs;
        goto 100;
    end,
end;

```

200:

```

for k:=1 to z do
begin
    u:=u+1;
    find_stars;
    find_sets_m_and_l;
    wj:=[];
    for k2:=1 to z do
    begin
        for j:=1 to y do
            if ((j in l[k]) and (j in l[k2])) then
                wj:=wj+l[k2];
        end,
        find_net_benefit,
        if (net_benefit>0) then
        begin
            iterations:=iterations+1;
            update_costs;
            goto 200;
        end,
    end;
end;

writeln('Heuristic Terminates here' );

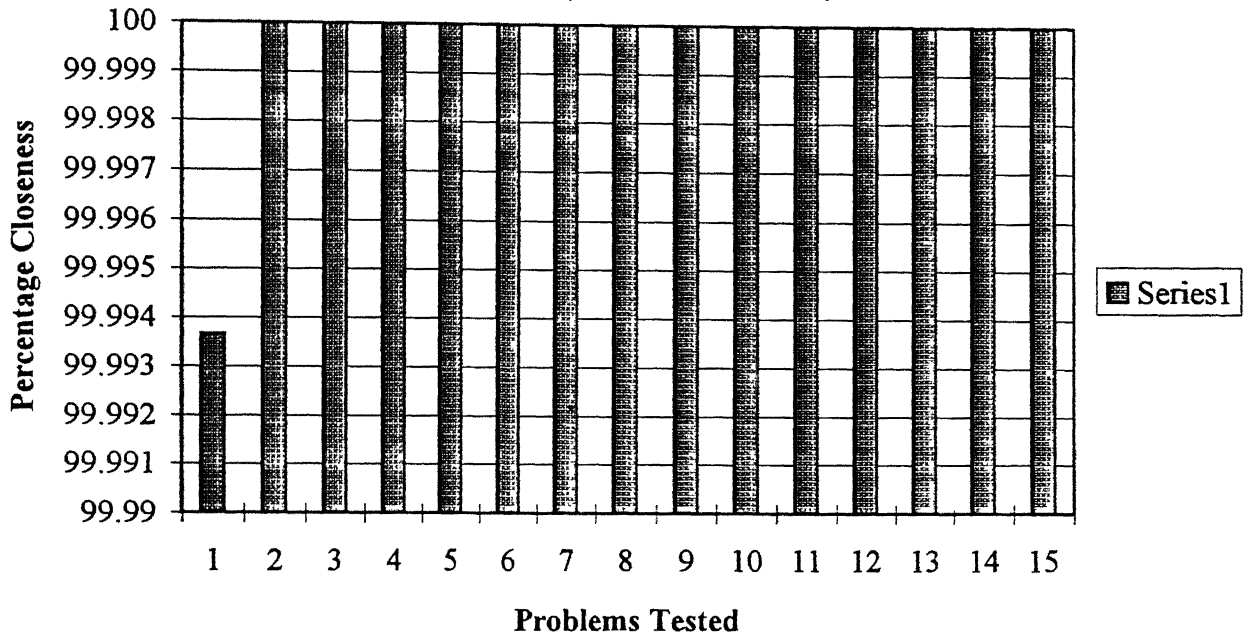
```

```
write('Final Optimal Solution is as follows :');  
writeln(opt);  
write('No. of Iterations taken by heuristics : ');  
writeln(iterations);  
writeln;
```

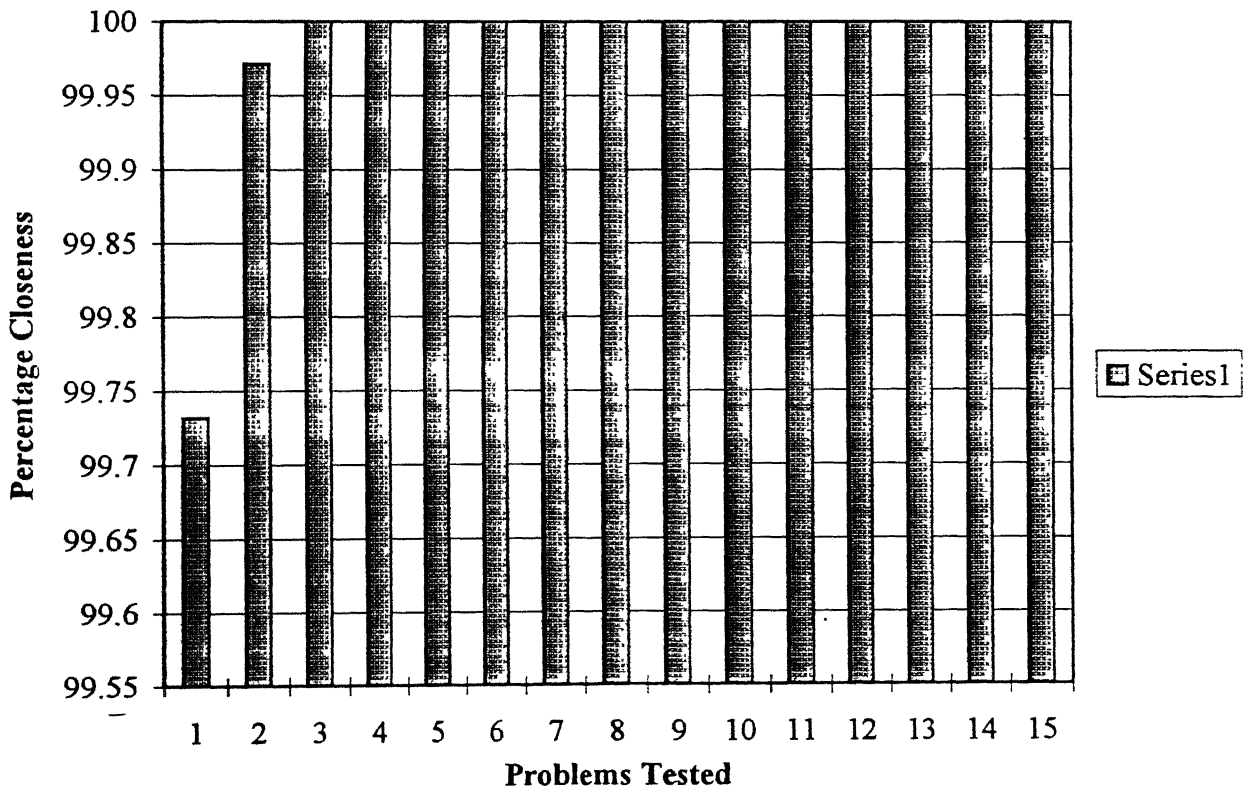
End.

Main Program Ends Here

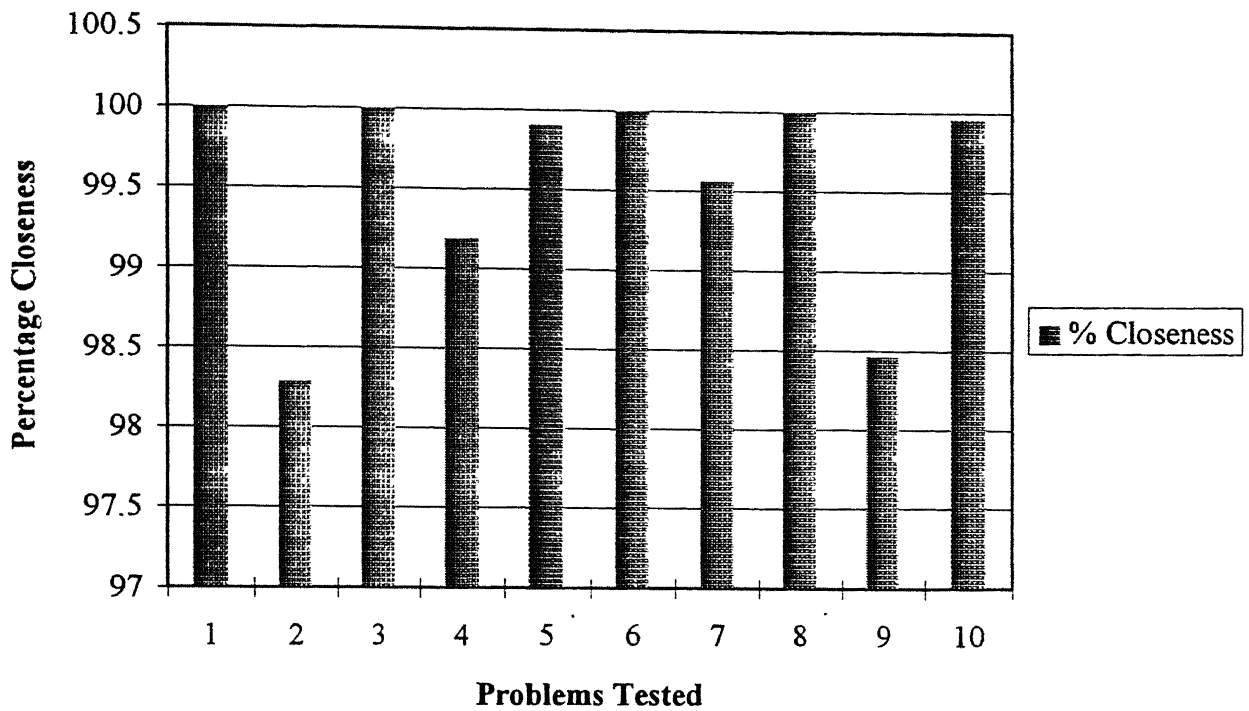
Problem Size : 3 x 3 x 3



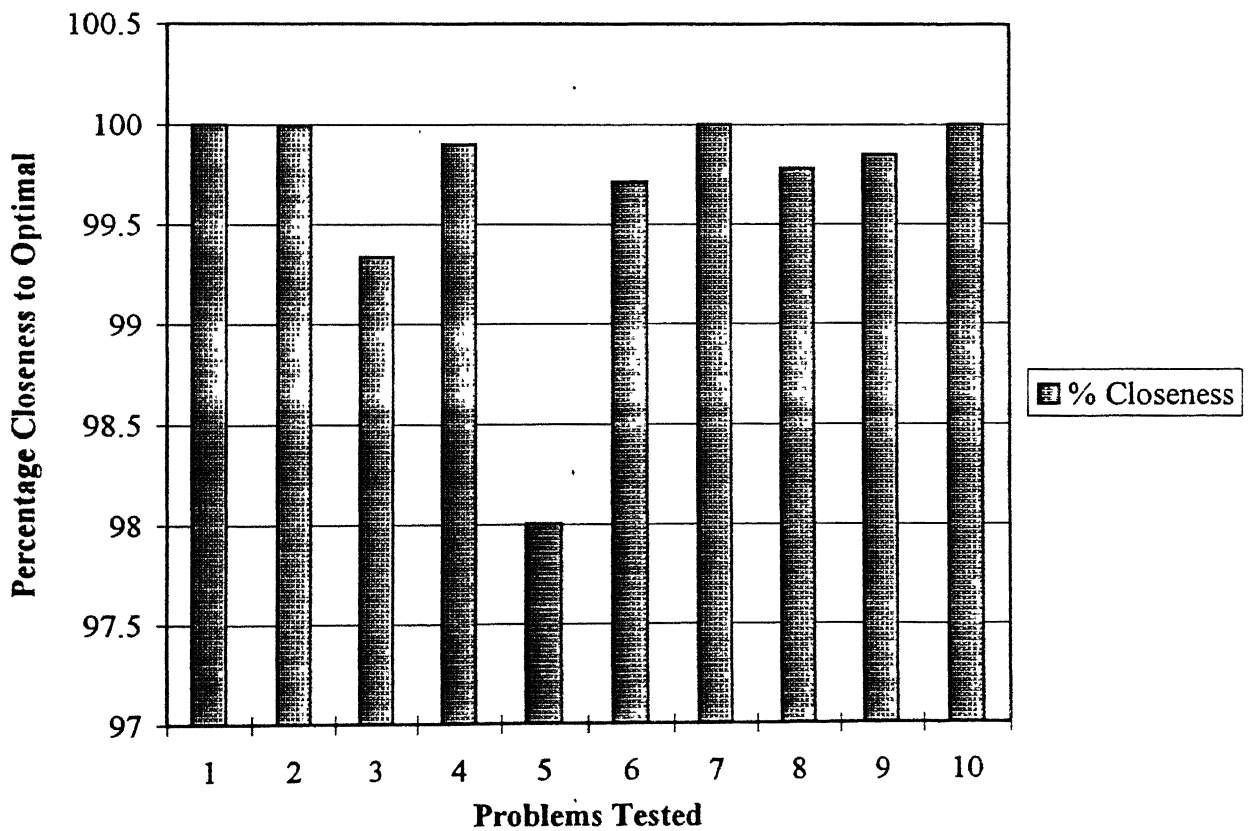
Problem Size : 4 x 4 x 4



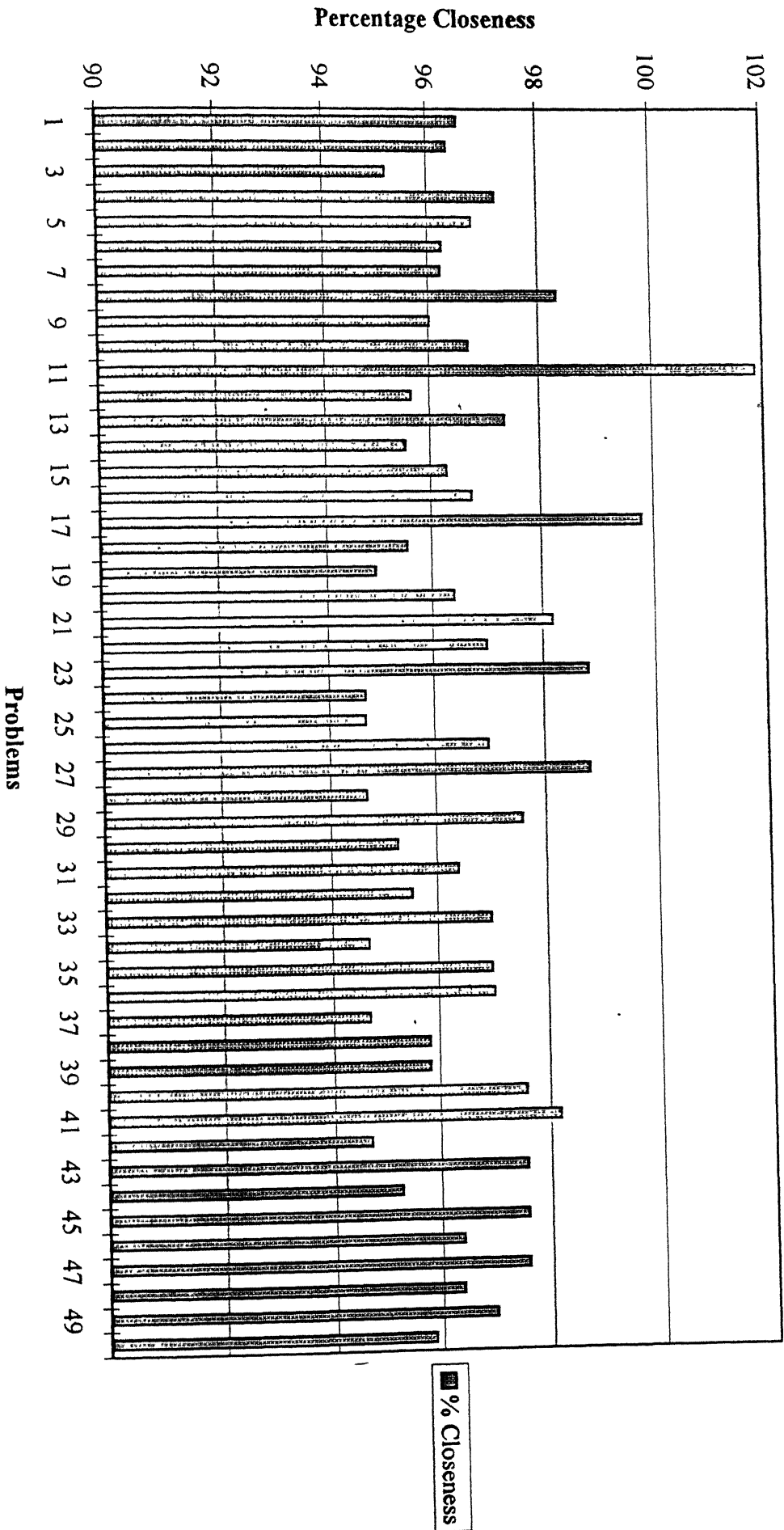
Problem Size : 5 x 5 x 5



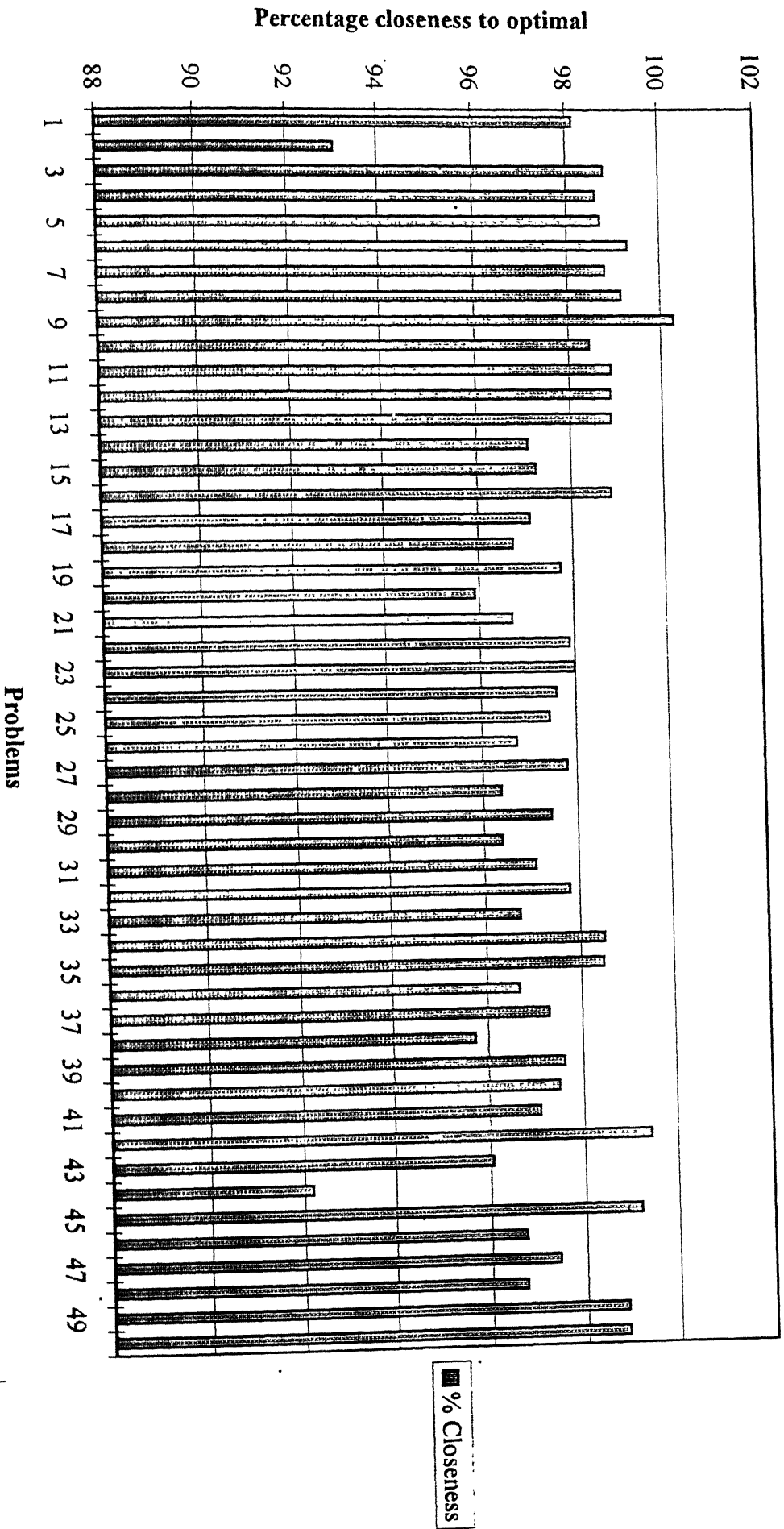
Problem Size : 7 x 7 x 7



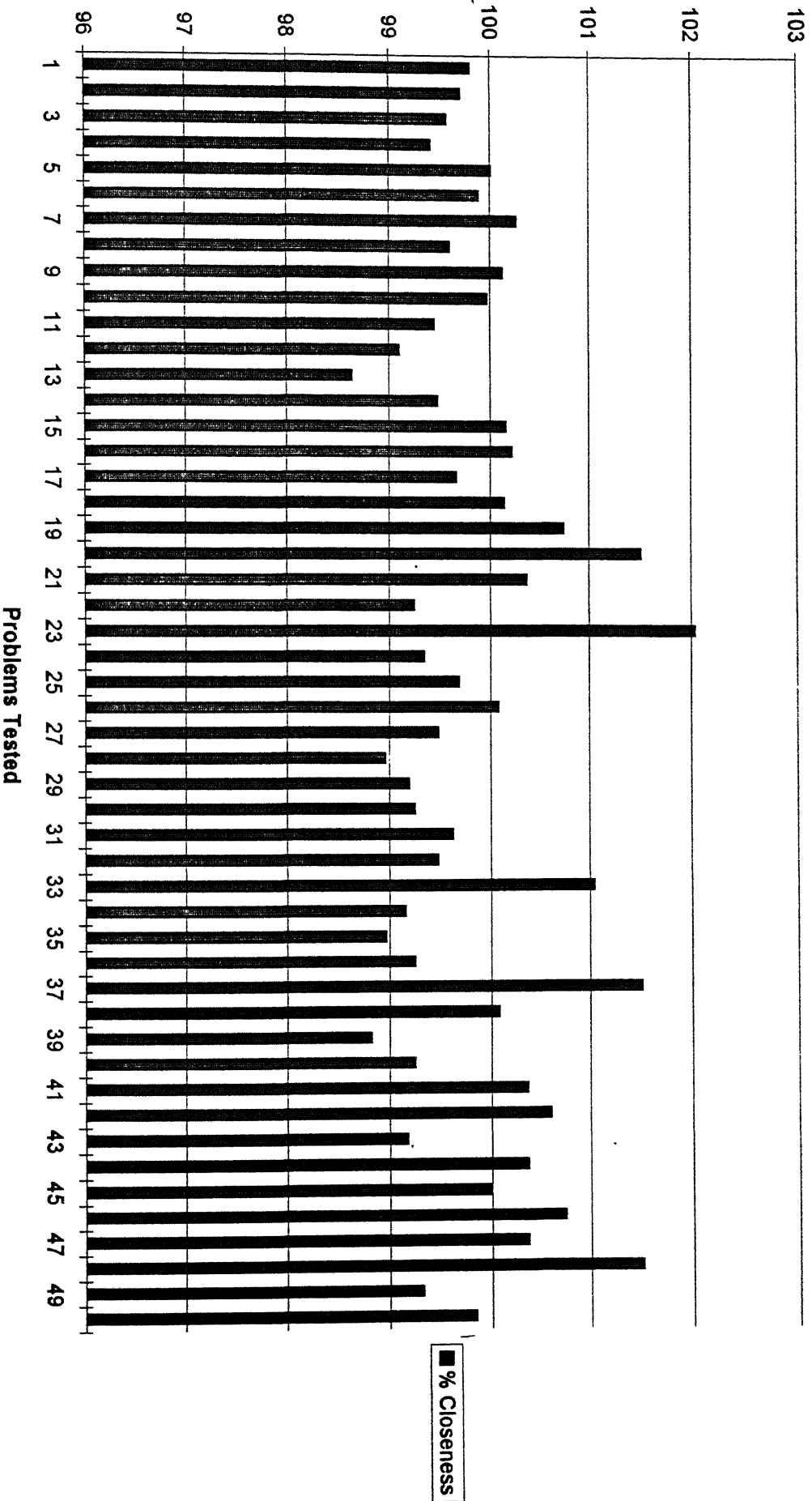
Problem Size : 100 x 100 x 100



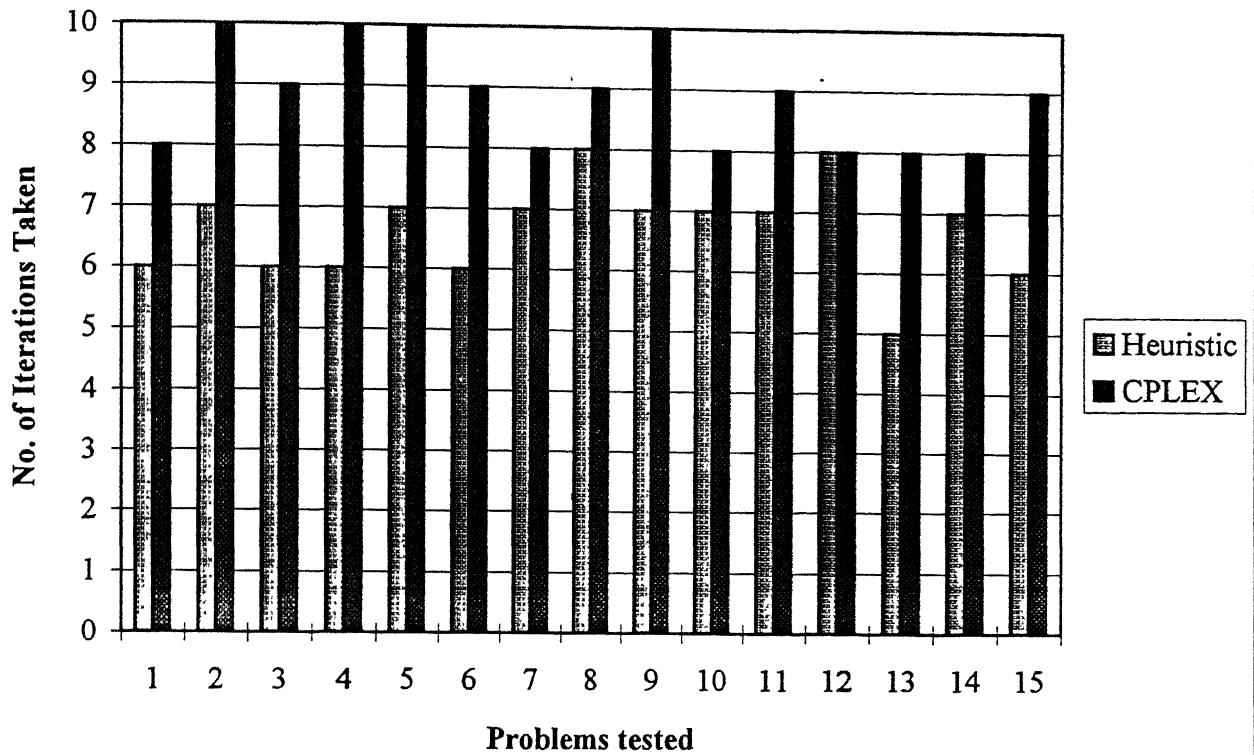
Problem Size : 150 x 150 x 150



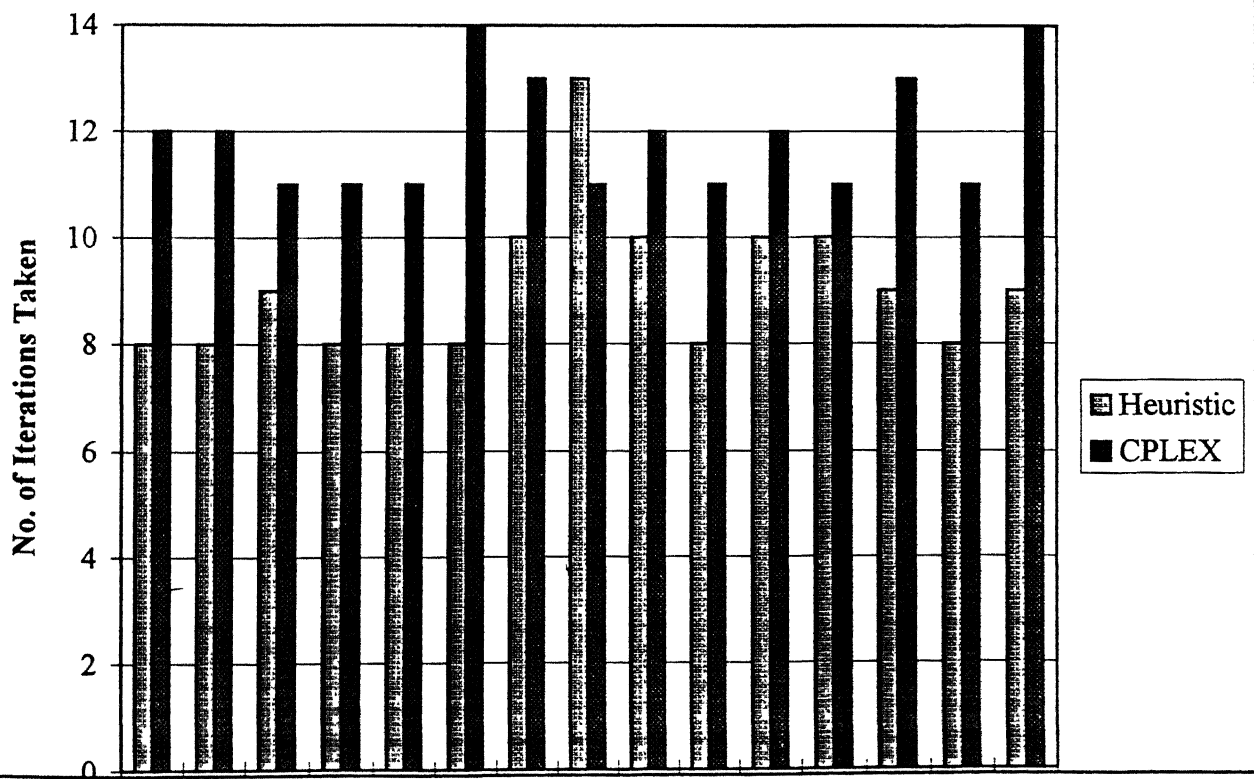
Problem Size : 200 x 200 x 200



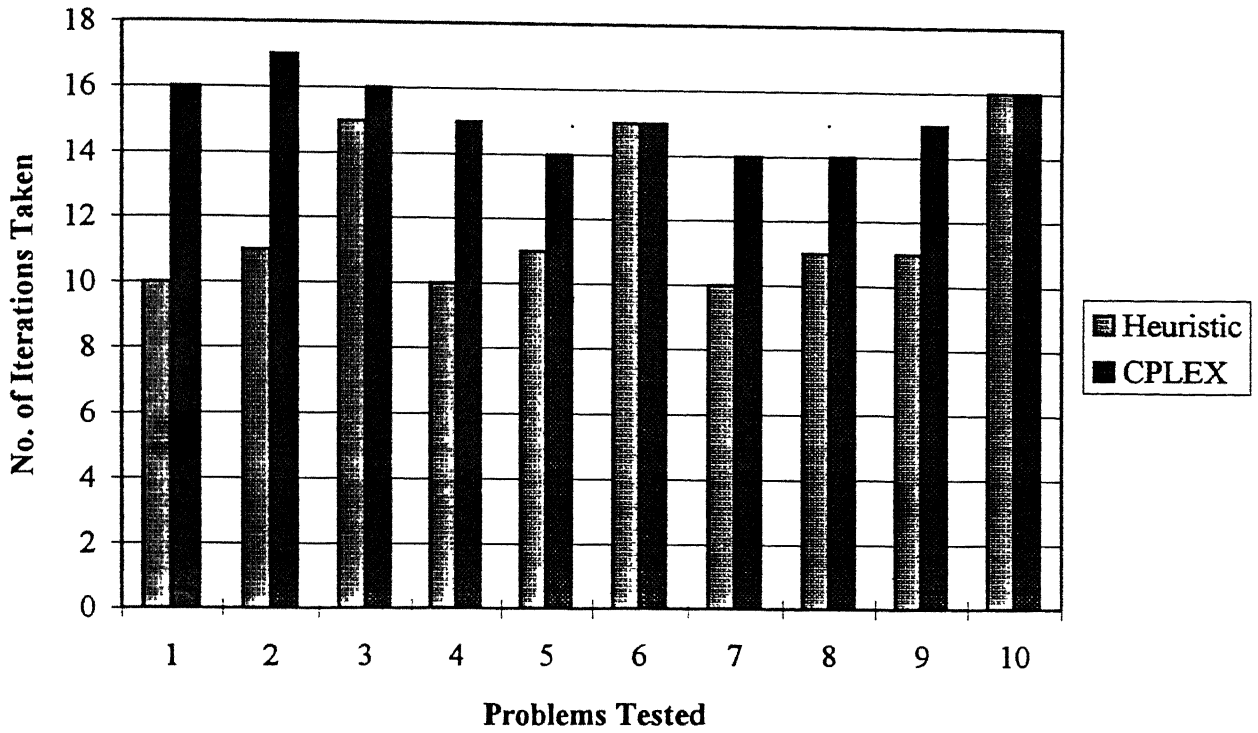
Problems Size : 3 x 3 x 3



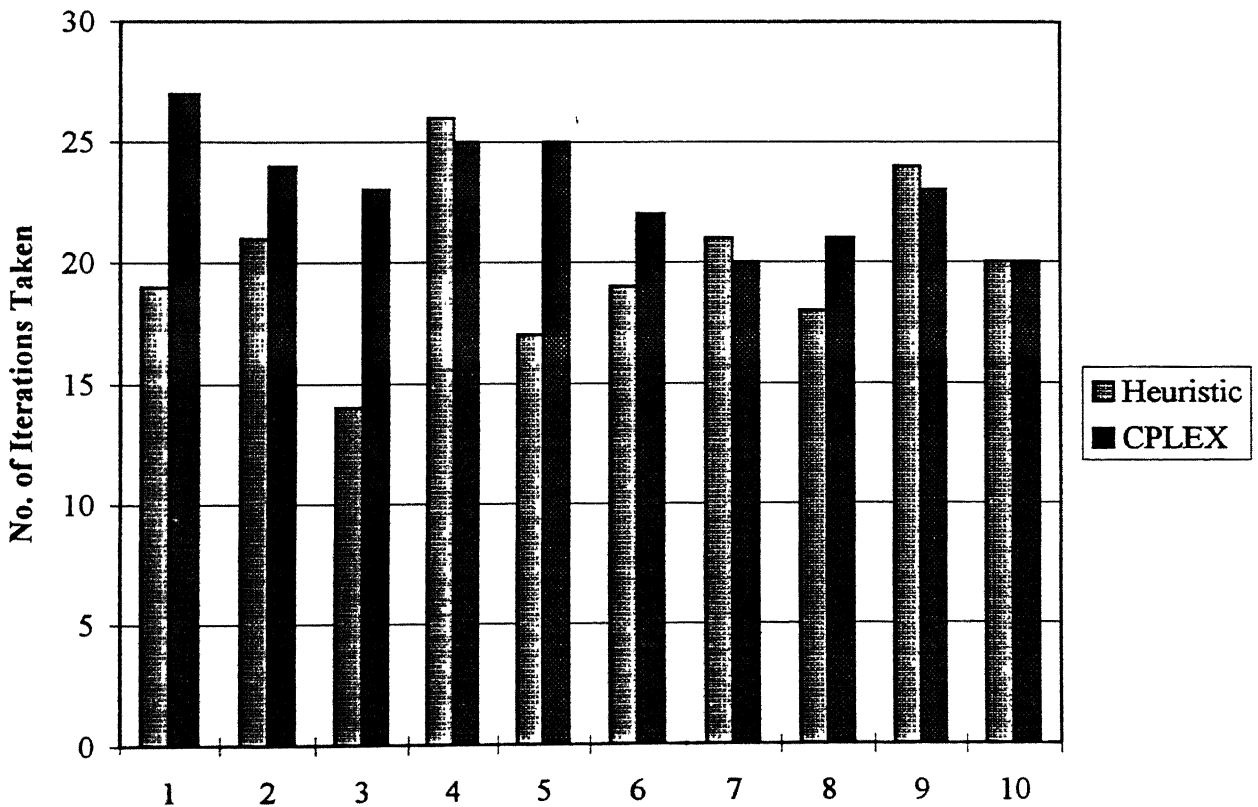
Problem Size : 4 x 4 x 4



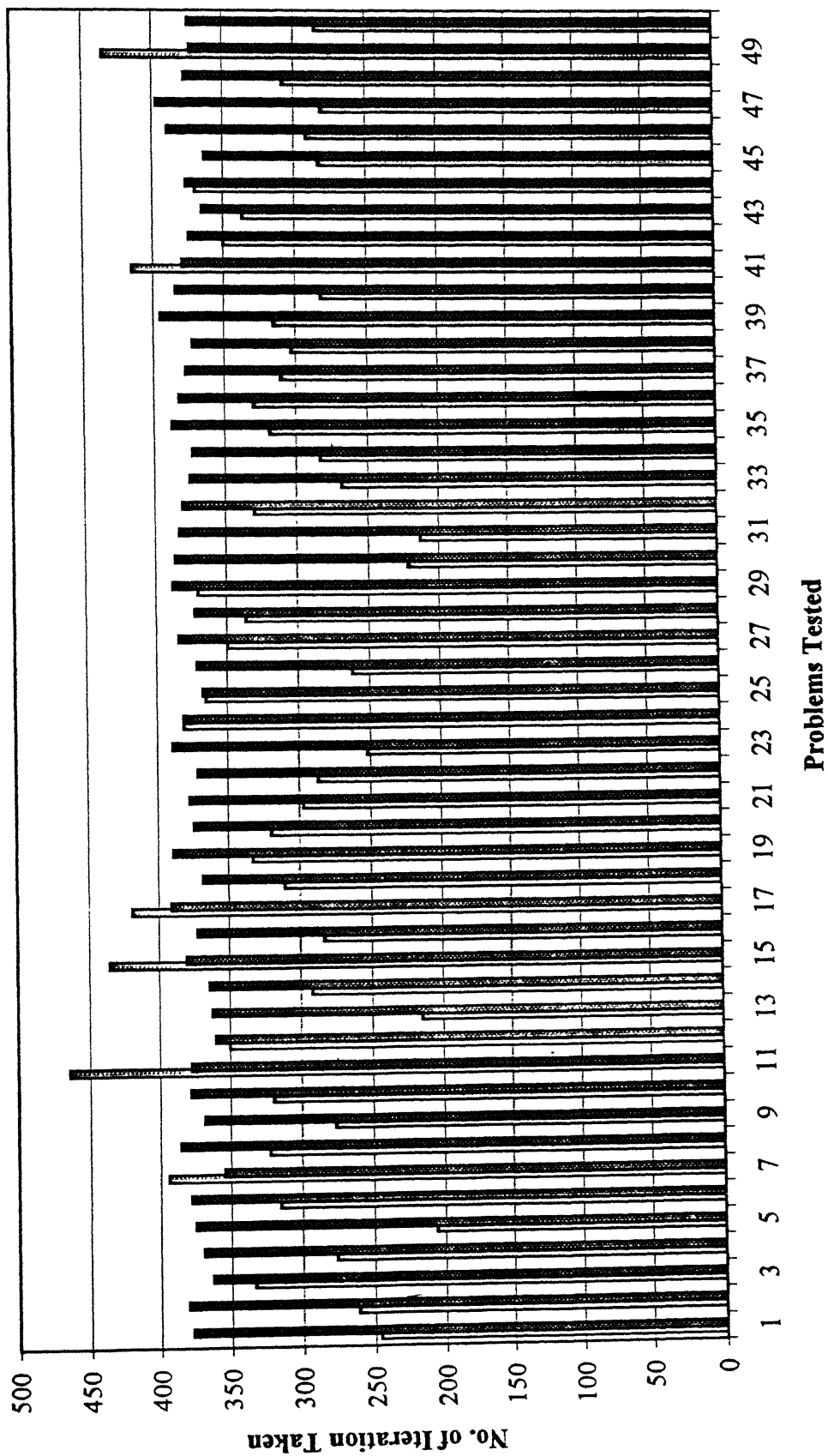
Problem Size : 5 x 5 x 5



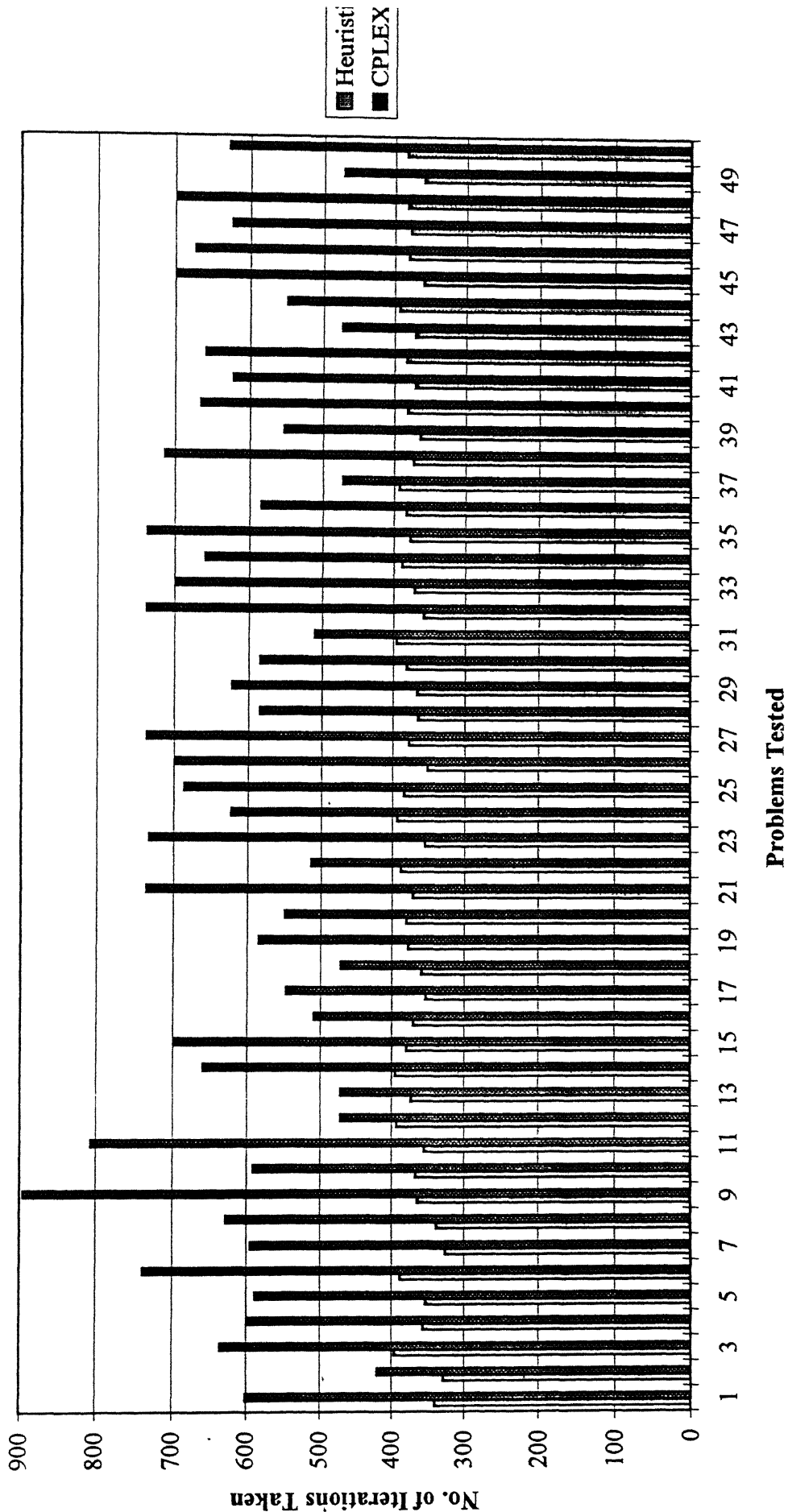
Problem Size : 7 x 7 x 7



Problem Size : 100 x 100 x 100



Problem Size : 150 x 150 x 150



Problem Size : 200 x 200 x 200

